

Supplementary Material for “Token-Level Self-Play with Importance-Aware Guidance for Large Language Models”

A Background

We consider a Large Language Model (LLM) parameterized by parameters θ , and denote its output distribution as π_θ . Given an input sequence x , commonly known as the prompt, the model generates a response sequence y . In preference-based alignment, the training dataset is made up of triplets $\mathcal{D} = \{(x, y_w, y_l)\}$, where y_w and y_l are two possible responses. Among them, y_w is considered better than y_l , which we write as $y_w \succ y_l$. We refer to y_w as the chosen (winning) response and y_l as the rejected (losing) one. In the following sections, we provide a brief overview of DPO, SPIN, recent token-level extensions of DPO and traditional knowledge distillation approaches.

Direct Preference Optimization (DPO). DPO [1] provides an elegant and efficient alternative to RLHF [2] by avoiding explicit reward model training. Instead, it reformulates the reward function $r(x, y)$ using a closed-form expression based on the ratio between the policy model and a fixed reference model:

$$r(x, y) = \beta \log \frac{\pi_\theta(y | x)}{\pi_{\text{ref}}(y | x)} + \beta \log Z(x), \quad (1)$$

where π_θ is the current policy model, π_{ref} is a static reference policy, and $Z(x)$ is a partition function independent of the policy. By plugging this reward into the Bradley-Terry framework [3], the preference probability between two responses y_w and y_l is modeled as $p(y_w \succ y_l | x) = \sigma(r(x, y_w) - r(x, y_l))$, where $\sigma(\cdot)$ denotes the sigmoid function. This leads to the DPO loss function, which optimizes the policy directly using preference data:

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right], \quad (2)$$

This formulation enables the model to directly incorporate human preference signals into policy learning without needing an explicit reward model or reinforcement learning.

Self-Play Fine-Tuning. Inspired by Generative Adversarial Networks (GAN) [4], Self-Play fine-tuning (SPIN) [5] proposes an iterative self-play framework where a language model fine-tunes itself by generating synthetic responses and learning to distinguish them from human-annotated data. Specifically, at each iteration t , the current model π_{θ_t} generates a response y' for a given prompt x , forming a synthetic preference pair $(x, y \succ y')$, where y is the ground-truth response. The model is then updated by minimizing:

$$\mathcal{L}_{\text{SPIN}}(\pi_{\theta_{t+1}}, \pi_{\theta_t}) = \mathbb{E}_{x, y \sim p_{\text{data}}, y' \sim \pi_{\theta_t}} \left[\ell \left(\beta \log \frac{\pi_{\theta_{t+1}}(y | x)}{\pi_{\theta_t}(y | x)} - \beta \log \frac{\pi_{\theta_{t+1}}(y' | x)}{\pi_{\theta_t}(y' | x)} \right) \right], \quad (3)$$

where $\ell(\cdot)$ is a convex decreasing loss (e.g., logistic loss), and β is a scaling factor. This formulation encourages the updated model to assign higher likelihood to responses resembling the ground-truth and lower likelihood to its own earlier responses. This self-play process eliminates the need for explicit reward models or preference-labeled data.

Token-Level DPO. Recent works have recognized the value of fine-grained feedback. Rafailov et al. [1] theoretically demonstrate that DPO can represent any dense reward function by reparameterizing it as an optimal advantage function within a token-level Markov Decision Process. This formulation enables DPO to effectively optimize policies at the token level. Building upon this, TDPO [6] introduces forward Kullback-Leibler divergence constraints and leverages the Bradley-Terry model to convert sentence-level preferences into token-level rewards, allowing the model to adjust its strategy dynamically during generation. Furthermore, TIS-DPO [7] enhances this approach by estimating token importance weights based on differences in prediction probabilities from contrastive language models, enabling importance sampling that approximates the optimal distribution by assigning weights to each token according to its estimated reward.

Traditional Knowledge Distillation Approaches. Knowledge distillation (KD) is a widely used technique to transfer knowledge from a larger teacher model to a smaller student model. Traditionally, KD employs the Kullback-Leibler (KL) divergence to minimize the difference between the teacher and student probability distributions [8]). Given a sequence \mathbf{x} , the student model learns to match the teacher’s output distribution by minimizing the following loss: $\mathcal{L}_{\text{KD}} = \sum_i D_{\text{KL}}(p(x_i | \mathbf{x}_{<i}, \tau) \| q_\theta(x_i | \mathbf{x}_{<i}, \tau))$, where $D_{\text{KL}}(\cdot \| \cdot)$ denotes the KL divergence and τ is the temperature to control the smoothness of the distributions.

B Implementation Details and Algorithm

B.1 Experiments Setup

We conduct two experimental settings to evaluate SWIFT: **Alignment** and **Knowledge Distillation**. The Alignment setting compares SWIFT with existing alignment methods, while the Knowledge Distillation setting assesses its effectiveness against existing knowledge distillation methods.

For the **Alignment** setting, we use Qwen1.5-1.8B [9] as the base model. As the teacher model, we adopt Zephyr-7B-SFT-Full [10], which is based on Mistral-7B [11] and further fine-tuned on the Ultrachat200k dataset¹ provided by HuggingFace. Ultrachat200k is a curated 200k subset of the UltraChat corpus [12], which consists of approximately 1.4 million high-quality instructional dialogues generated via OpenAI’s Turbo API. We follow the procedure in [5], 50,000 prompts are randomly sampled from Ultrachat200k and generate synthetic responses using the base model. For evaluation, we adopt the HuggingFace Open LLM Leaderboard [13], implemented via the Language Model Evaluation Harness [14]. The leaderboard covers six representative benchmarks targeting different capabilities of LLMs: commonsense reasoning (ARC [15], HellaSwag [16], Winogrande [17]), multi-task language understanding (MMLU [18]), resistance to misinformation (TruthfulQA [19]), and mathematical reasoning (GSM8k [20]). These benchmarks collectively provide a rigorous and diverse framework for evaluating both alignment quality and generalization. Additionally, we evaluated the output quality of the LLM using MT-bench [21] with its provided dataset, we use the API of GPT-4 as the judge. The details of the benchmarks are provided in table 1 below.

Table 1: Details of the HuggingFace Open LLM Leaderboard evaluation datasets, including the number of few-shot examples and the evaluation metric for each.

Dataset	Arc	TruthfulQA	Winogrande	GSM8k	HellaSwag	MMLU
# Few-shot	25	0	5	5	10	5
Metric	acc_norm	mc2	acc	acc	acc_norm	acc

For the **Knowledge Distillation** setting, We use GPT2-1.5B [22] as the base model and Qwen2.5-7B-Instruct [23] as the teacher. Four datasets are selected for evaluation: DATABRICKSDOLLY-15K (**Dolly**) [24], ALPACA (**Alpaca**) [25], S-NI (**S-NI**) [26], and DIALOGSUM (**Dialogsum**) [27]. Final performance is reported using the ROUGE-L metric [28] between generated outputs and human-annotated references. In the state-of-the-art method DSKD [29], distillation is typically performed on a single dataset, while evaluation is conducted across multiple datasets spanning different domains or tasks. In contrast, we construct separate training, validation, and testing splits for each domain, allowing for a more targeted evaluation of knowledge distillation within the same domain. The details of the datasets are provided in table 2 below.

Table 2: Dataset Statistics

Dataset	Train	Validation	Test
Dolly	11,435	1,000	500
Alpaca	10,396	500	500
S-NI	10,414	500	1,902
DialogSum	12,460	500	1,500

¹https://huggingface.co/datasets/HuggingFaceH4/ultrachat_200k

77 B.2 Hyperparameters

78 To reduce training costs and memory consumption, we employ DeepSpeed ZeRO-3 [30] and
 79 FlashAttention-2 [31] throughout all training iterations. Models are trained using the RMSProp
 80 optimizer [32] without weight decay, following standard practice for LLM alignment fine-tuning.
 81 We set the global batch size to 2, use bfloat16 precision, and apply a 10% linear warmup at the start
 82 of each iteration. The peak learning rate is set to 5×10^{-7} for iterations 0 and 1, and 1×10^{-7} for
 83 iterations 2 and 3 as training approaches convergence. Each iteration is trained for 2 epochs with a
 84 maximum sequence length of 2048 tokens. For token importance estimation as defined in equation
 85 14 in main paper, we set $\mu = 1$, with lower and upper clipping bounds $L = -0.5$ and $U = 1.5$,
 86 respectively. The hyperparameter k is fixed to 1. All experiments are conducted on $2 \times$ NVIDIA RTX
 87 4090 GPUs.

88 B.3 Synthetic Data Generation

89 We generate synthetic rejected responses using the library vLLM [33] to speed up inference
 90 with distributed inference over multiple GPUs. We use a sampling decoding strategy to gener-
 91 ate responses, with a temperature of 1.0 and top_p of 1.0. We consider the prompting template
 92 `\n\n<Human>:{prompt}\n\n<Assistant>:.`

93 B.4 Algorithm

94 We provided the pseudocode of our proposed **SWIFT (Self-Play Weighted Fine-Tuning)** method in
 95 Algorithm 1.

Algorithm 1 Self-Play Weighted Fine-Tuning (SWIFT)

Require: $\{(x_i, y_i)\}_{i \in [N]}$: SFT dataset, π_{θ_0} : LLM with initial parameters θ_0 , π_{θ_T} : teacher model,
 M : number of iterations.
 1: **for** $t = 0, \dots, M - 1$ **do**
 2: Compute token importance weights ω using π_{θ_T} and π_{θ_t}
 3: **for** $i = 1, \dots, N$ **do**
 4: Generate synthetic data $y'_i \sim \pi_{\theta_t}(\cdot | x_i)$.
 5: **end for**
 6: $\theta_{t+1} = \arg \min_{\theta \in \Theta} \sum_{i \in [N]} [l(u(x, y, y', \pi_{\theta^S}, \omega) - v(x, y, y', \pi_{\theta^S}, \omega))]$
 7: **end for**
 8: **return** θ_T .

96 B.5 Details of Teacher-Guided Token Importance Estimation

97 In this section, we provide a detailed explanation of the method described in Section 3.3 (Teacher-
 98 Guided Token Importance Estimation) of the main paper, addressing how token-level importance
 99 weights are distilled from a teacher model and mapped to student tokens, especially in the presence
 100 of tokenizer mismatches. We also include the full algorithm, computational analysis, and illustrative
 101 examples to enhance clarity.

102 B.5.1 Overview of the Method

103 Our aim is to develop an efficient mechanism to estimate the importance weight of each token.
 104 Intuitively, a well-trained teacher—having been trained on more data with higher model capacity—can
 105 better signal which tokens carry higher “reward”. In this section, we present a principled method to
 106 distill token-level reward signals from a teacher model θ^T and transfer them to a student model θ^S .
 107 Inspired by TIS-DPO [7], we first estimate the raw importance weight for the t^{th} token in a response
 108 $y = (y_1, \dots, y_T)$ as

$$\omega_t^{\text{raw}} = k \cdot \exp\left(\mu \cdot \text{clamp}\left(\log \frac{\pi_{\theta^T}(y_t | x, y^{<t})}{\pi_{\theta^S}(y_t | x, y^{<t})}, L, U\right)\right), \quad (4)$$

where $\text{clamp}(a, L, U) = \min(\max(a, L), U)$ truncates the log-odds to $[L, U]$ to reduce variance and stabilize optimization. Here $k > 0$ and μ are constant; L and U are lower and upper clipping bounds.

However, this formulation assumes that the teacher and student models share the same tokenizer and vocabulary. In practice, this assumption rarely holds. A sequence y might be tokenized by the teacher model as $[y_1^T, y_2^T, \dots, y_\ell^T]$, and by the student as a different sequence $[y_1^S, y_2^S, \dots, y_s^S]$, making direct token-to-token alignment in (4) infeasible. To address the misalignment issue caused by differing tokenizers, we propose a generalizable token mapping strategy based on shared surface segments.

Case 1: Shared Tokenizer. In the scenario where both the teacher and student models utilize an identical tokenizer, the alignment between their respective token sequences is inherently preserved. Specifically, each token y_t generated by the student model corresponds directly to a token produced by the teacher model at the same position, thereby eliminating the need for additional alignment mechanisms. As a result, the distillation weight ω_t associated with each token position t can be directly adopted from the unprocessed or raw weighting scheme. Formally, we define:

$$\omega_t = \omega_t^{\text{raw}}, \quad (5)$$

where ω_t^{raw} represents the original distillation weight derived from the teacher model’s outputs as in 4 without any post-processing or re-alignment.

Case 2: Different Tokenizers. When tokenizers differ, direct index alignment between tokens becomes unreliable. Rather than aligning tokens individually, we instead leverage a common structural unit shared across both tokenizations. Most modern tokenization algorithms, including BPE [34], SentencePiece [35], and WordPiece [36], segment text into subwords while generally respecting word boundaries. This means that each token strictly belongs to either a single word or the whitespace separating words, making word spans a natural and reliable basis for alignment. Leveraging this property, we segment the original text into coarse-grained units corresponding to **complete words and their leading whitespace**, resulting in a sequence of shared components $\{c_1, c_2, \dots, c_K\}$. This ensures that every token from both teacher and student sequences can be assigned to one of these shared components, which are small enough to preserve local semantics but broad enough to encompass all subword variations.

Suppose student token y_i^S belongs to component c_h , and the teacher tokens in that segment are $T(c_h) = \{y_j^T, y_{j+1}^T, \dots, y_{j+\ell}^T\}$. Assuming all tokens within a shared component contribute equally, we define the student’s token importance as the average of the teacher tokens’ weights in the corresponding component c_h :

$$\omega_i^S = \frac{1}{|T(c_h)|} \sum_{y_r^T \in T(c_h)} k \cdot \exp\left(\mu \cdot \text{clamp}\left(\log \frac{\pi_{\theta^T}(y_r^T | x, y_{<r}^T)}{\pi_{\theta^S}(y_i^S | x, y_{<i}^S)}, L, U\right)\right). \quad (6)$$

B.5.2 Pseudo-Code and Computational analysis for Teacher-Guided Token Weight Estimation

The complete process of the Teacher-Guided Token Weight Estimation method is described in Algorithm 2.

Given a text response y , we first tokenize it using the teacher tokenizer tok_T and the student tokenizer tok_S to obtain token sequences Y^T and Y^S , respectively. In addition, the raw text y is segmented into a sequence of shared components \mathcal{C} , where each component consists of a full word along with its leading whitespace. This segmentation **has a time complexity of $\mathcal{O}(n)$** , where n is the number of characters in the input y .

For each component $c \in \mathcal{C}$, we define $T(c)$ and $S(c)$ as the sets of teacher and student tokens that belong to c . We then will estimate the weight for all the student tokens belonging to this component c . Since each token belongs to exactly one component, **the algorithm processes each student token only once**.

The core computation involves measuring the discrepancy between the teacher and student log-likelihoods. For each student token $i \in S(c)$, we compute the log-probability under the student model, and for each teacher token $t \in T(c)$, we compute its log-probability under the teacher

Algorithm 2 Teacher-Guided Token Importance Estimation

Require: Teacher model π_{θ_T} , student model π_{θ_S} , teacher tokenizer tok_T , student tokenizer tok_S , hyperparameters (k, μ, L, U) , response y .

Ensure: Importance weights $\omega_{1:|y^S|}^S$ for student tokens.

```
1:  $Y^T \leftarrow \text{tok}_T(y)$ ,  $Y^S \leftarrow \text{tok}_S(y)$ 
2:  $\mathcal{C} \leftarrow \text{WORDSEGMENTS}(y)$  ▷ Segment raw text into word-level spans
3: for each component  $c$  in  $\mathcal{C}$  do
4:    $T(c) \leftarrow \{t : Y_t^T \subset c\}$ ,  $S(c) \leftarrow \{i : Y_i^S \subset c\}$ 
5:   for each  $t \in T(c)$  do
6:      $a_t \leftarrow \log(\pi_{\theta_T}(Y_t^T \mid x, Y_{<t}^T))$ 
7:   end for
8:   for each  $i \in S(c)$  do
9:      $b_i \leftarrow \log(\pi_{\theta_S}(Y_i^S \mid x, Y_{<i}^S))$ 
10:    for each  $t \in T(c)$  do
11:       $\ell_{t,i} \leftarrow a_t - b_i$ 
12:       $\omega_t^{\text{raw}} \leftarrow k \cdot \exp(\mu \cdot \text{clamp}(\ell_{t,i}, L, U))$ 
13:    end for
14:     $\omega_i^S \leftarrow \frac{1}{|T(c)|} \sum_{t \in T(c)} \omega_t^{\text{raw}}$  ▷ eq 6
15:  end for
16: end for
17: return  $\{\omega_i^S\}_{i=1}^{|Y^S|}$ 
```

156 model. The weight for student token i is then obtained by averaging the clamped exponential of the
157 log-probability differences between the teacher and student tokens within the same component, as
158 formalized in Equation 6.

159 This design has two critical efficiency advantages: (i) the decoupled log probability computation
160 allows **for only a single forward pass** per model per response, and (ii) the local averaging over
161 shared components yields robust token importance estimates. These properties make the algorithm
162 both scalable and robust to differences in tokenization schemes between teacher and student models.

163 B.5.3 Illustrative Examples

164 Figure 1 illustrates our Teacher-Guided Token Weight Estimation method applied to the input
165 "Taylor Swift is a singer not an AI algorithm". The sentence is segmented into shared
166 components (green), which serve as alignment components between teacher tokens (red) and student
167 tokens (blue). For each component, token-level log-probabilities are computed from both models. Stu-
168 dent token weights are then computed using a divergence function $D(x) = k \cdot \exp(\mu \cdot \text{clamp}(x, L, U))$
169 over the difference between teacher and student log-probabilities. In cases where multiple tokens
170 align to the same component, weights are averaged across the corresponding pairs.

171 Note that, there may be more than one whitespace character between two words, or including newline
172 characters (e.g., `\n`). Importantly, modern tokenizers do not tokenize words alone—they also treat
173 sequences of whitespace (spaces, tabs, or newlines) as separate tokens. In our implementation, we
174 preserve these segments as standalone components, denoted as `[SPACE]`.

175 B.5.4 Comparison with Prior Work.

176 Compared to the contrastive-based token weighting in [7], our method is significantly more computa-
177 tionally efficient, as it does not require training separate positive and negative models. We validate
178 this by providing a direct comparison of the total time cost between SWIFT and the contrastive-based
179 token weighting approach (TIS-DPO) in Appendix C.1. Furthermore, as shown in Table 3 in main
180 paper, our teacher-guided estimation consistently achieves superior performance, confirming both its
181 efficacy and practicality.

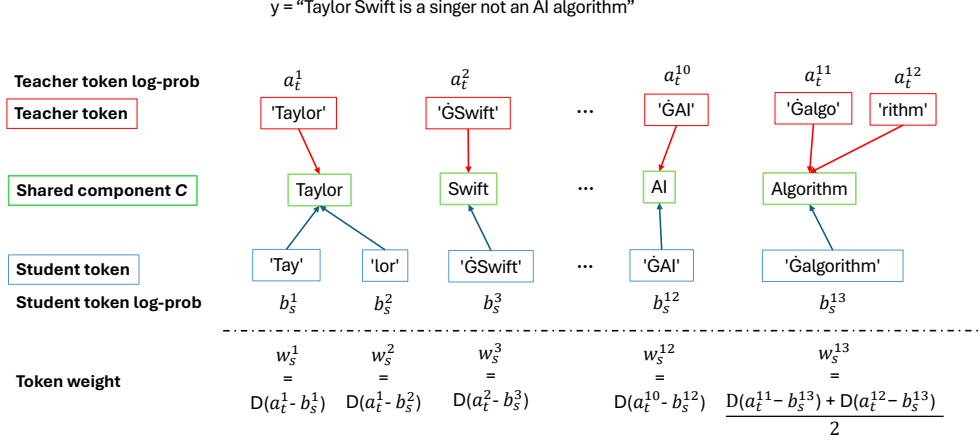


Figure 1: An example of our Teacher-Guided Token Weight Estimation method where $D(x) = k \cdot \exp(\mu \cdot \text{clamp}(x, L, U))$

C Additional Experiment Result

C.1 Training Overhead

To further examine the computational efficiency of our method, we compare the total runtime of the full training pipeline between SWIFT and the contrastive-based token weighting approach (TIS-DPO), as introduced in [7]. The TIS-DPO framework comprises three strategies for estimating token weights: (1) TIS-DPO (P), which guides the original LLM with contrastive prompts; (2) TIS-DPO (S), which involves training two separate LLMs via SFT on winning and losing responses, respectively; and (3) TIS-DPO (D), which performs both forward and reverse DPO training using winning and losing responses. Among these, [7] report that TIS-DPO (D) achieves the strongest empirical performance, and thus we adopt it for our comparison.

Table 3: Total times for different iterations under SWIFT and TIS-DPO compute weight method.

Iteration	SWIFT			TIS-DPO (D)		
	Generation	Compute Weight	Training	Generation	Compute Weight	Training
Iter 0	0.52h	3.08h	6.03h	0.52h	15.33h	6.16h
Iter 1	0.51h	3.05h	6.13h	0.52h	15.24h	6.10h
Iter 2	0.52h	3.10h	6.01h	0.50h	15.15h	6.08h
Iter 3	0.49h	3.05h	6.10h	0.51h	15.28h	6.06h

Table 3 presents the generation, weight computation, and training times for both SWIFT and TIS-DPO (D) across different self-play iterations. Notably, while generation and training times are comparable between the two approaches, SWIFT demonstrates a significant reduction in weight computation time. This efficiency gain arises because TIS-DPO (D) requires training two distinct models to obtain token weights. This highlights the superior computational efficiency of our approach over TIS-DPO (D). Moreover, as shown in Table 3 in main paper, SWIFT not only offers reduced overhead but also achieves better overall performance.

C.2 Generation Examples

Tables 4, 5, and 6 present additional generation examples produced by our SWIFT fine-tuned model across different training iterations. Table 4 showcases examples from math problems, Table 5 focuses on code-generation tasks, and Table 6 includes general instruction-following cases. These examples illustrate a clear improvement in response quality compared to the outputs from the initial

204 SFT checkpoint. Notably, as the number of iterations increases, the model’s responses become
 205 more concise and better aligned with the ground-truth completions. For clarity, in each prompt,
 206 the responses generated by SFT, iteration 0, iteration 1, and iteration 2 are respectively labeled as
 207 **Response 1**, **Response 2**, **Response 3**, and **Response 4**.

Table 4: Generation example of our fine-tuned model by SWIFT at different iterations. GPT-4 chooses SWIFT iteration 2 over SFT, and iteration 0, 1.

Prompt	<Human>:If the year is divisible by both 4 and 100, it is not a leap year, unless it is also divisible by 400. <Assistant>:
Ground truth	This is a true statement. For example, the year 1900 was divisible by both 4 and 100 but not by 400, so it was not a leap year. However, the year 2000 was divisible by both 4 and 100 and also by 400, so it was a leap year.
SFT	This is true. For example, the year 1900 was divisible by both 4 and 100 but not by 400, so it was not a leap year. However, 1904 was divisible by 4, but not by 100, so it was a leap year. The same applies to the year 2000, 2004, and 2008.
Iteration 0	True. For example, the year 1900 was divisible by both 4 and 100 but not by 400, so it was not a leap year. However, the year 2000 was divisible by both 4 and 100 and also by 400, so it was a leap year.
Iteration 1	This is a true statement. For example, the year 1900 was divisible by both 4 and 100 but not by 400, so it was not a leap year. However, the year 2000 was divisible by both 4 and 100 and also by 400, so it was a leap year.
Iteration 2	This is a true statement. For example, the year 1900 was divisible by both 4 and 100 but not by 400, so it was not a leap year. However, the year 2000 was divisible by both 4 and 100 as well as 400, so it was a leap year.
Judgment	Comparison: Response 3 = Response 4 > Response 2 > Response 1. Explanation: Responses 3 and 4 are identical and fully match the ground truth both in correctness and clarity, providing the textbook explanation and appropriate examples (1900 and 2000) without unnecessary additions. Response 2 is also correct and concise but slightly less formal in tone by starting with "True" instead of a complete sentence, which slightly reduces its explanatory quality. Response 1, while factually correct in its individual statements, introduces unnecessary examples (1904, 2004, 2008) that deviate from the prompt’s focus on the specific exception rule involving divisibility by 100 and 400, potentially confusing the reader by adding standard leap year cases that do not illustrate the special condition described in the prompt.

Table 5: Generation example of our fine-tuned model by SWIFT at different iterations. GPT-4 chooses SWIFT iteration 2 over SFT, and iteration 0, 1.

Prompt	<p><Human>:Write a Python program that prompts the user to enter a string and then counts the number of characters in the entered string. The program should print the total number of characters in the string.</p> <p><Assistant>:</p>
Ground truth	<pre>String = input("Enter a string: ") count = 0 for character in string: count += 1 print("The total number of characters in the string is:", count)</pre>
SFT	<p>Here’s a possible solution:</p> <pre>string = input("Enter a string: ") count = 0 for character in string: count += 1 print("The total number of characters in the string is:", count) <p>Here’s how the program works:</p> <ol style="list-style-type: none"> 1. We prompt the user to enter a string using the input() function and store the string in the variable string. 2. We initialize a variable count to 0, which will be used to keep track of the number of characters in the string. 3. We use a for loop to iterate over each character in the string. The loop iterates over each character in the string using string[i], which gives us a slice of the string that starts at the current position and goes up to (but not including) the end of the string. We add 1 to the count variable to increment it for each character we encounter. 4. We use the print() function to output the result to the screen. We use string interpolation to embed the count variable in the output string, so it’s easy to see how many characters are in the string. 5. The program terminates when the user enters a string, so we use a while loop to wait for user input before printing the final result. </pre>
Iteration 0	<pre>String = input("Enter a string: ") count = 0 for character in string: count += 1 print("Total number of characters in the entered string is: ", count)</pre>
Iteration 1	<pre>String = input("Enter a string: ") count = 0 for char in string: count += 1 print("The total number of characters in the string is:", count)</pre>
Iteration 2	<pre>String = input("Enter a string: ") count = 0 for character in string: count += 1 print("The total number of characters in the string is:", count)</pre>
Judgment	<p>Comparison: Response 4 = Response 3 > Response 2 > Response 1.</p> <p>Explanation: Responses 4 and 3 are functionally identical to the ground truth, only differing slightly in variable naming (“char” vs. “character”) but maintaining correct, concise, and syntactically sound code. Response 2 is also correct but has a minor stylistic difference in the output string (“Total number of characters in the entered string is”) that deviates from the ground truth and slightly affects alignment. Response 1, while technically correct in the code block, contains multiple issues in the explanation: it introduces confusion by mentioning a nonexistent while loop and incorrectly describes the use of string[i], which is invalid syntax and misrepresents how the loop operates. These inaccuracies make it the least reliable response.</p>

Table 6: Generation example of our fine-tuned model by SWIFT at different iterations. GPT-4 chooses SWIFT iteration 2 over SFT, and iteration 0, 1.

Prompt	<p><Human>:Given the text: During The Daily Show’s “Between the Scenes” segment on April 10, Noah asked television mogul Winfrey if there was any sense of normalcy left in her life. The answer, quite simply, was no. Even when she tried to dig up a passable answer, she hilariously failed. The avocado orchard is tucked away on the property of Winfrey’s 23,000-square-foot Montecito home, which she purchased for \$50 million back in 2001. The home also boasts a 5,000-square-foot ranch-style house, a pool, horse stables, an equestrian ring, a fruit orchard, and more. Simply put, Oprah’s home — much like guacamole — is extra. Get it? Could you provide more information about the features of Oprah Winfrey’s Montecito property, including the size of her home and other amenities?</p> <p><Assistant>:</p>
Ground truth	Yes, Oprah Winfrey’s Montecito property is 23,000-square-foot in size and includes a 5,000-square-foot ranch-style house, a pool, horse stables, an equestrian ring, a fruit orchard, and an avocado orchard. The property was purchased for \$50 million in 2001.
SFT	Yes, the text mentions that Oprah Winfrey purchased her 23,000-square-foot Montecito home for \$50 million in 2001. It also states that the home has a 5,000-square-foot ranch-style house, a pool, horse stables, an equestrian ring, a fruit orchard, and an avocado orchard. Therefore, the size of Oprah’s home is not given in the text.
Iteration 0	According to the text, Oprah Winfrey purchased her Montecito property for \$50 million in 2001. The property consists of a 23,000-square-foot Montecito home and a 5,000-square-foot ranch-style house, an irrigation pond, natural stone walkways, a fountain, tailgating capability, and more. Additionally, the property includes a fruit orchard and an avocado orchard.
Iteration 1	Of course! Oprah Winfrey’s Montecito property is 23,000-square-foot in size and was purchased for \$50 million in 2001. The property also includes a 5,000-square-foot Ranch-style house, a pool, horse stables, an equestrian ring, a fruit orchard, and an avocado orchard. It is truly a sprawling estate that puts even avocado-curious minds to shame.
Iteration 2	Yes, Oprah Winfrey’s Montecito property is 23,000-square-foot in size and includes a 5,000-square-foot ranch-style house, pool, horse stables, an equestrian ring, a fruit orchard, and an avocado orchard. She purchased the property for \$50 million in 2001.
Judgment	<p>Comparison: Response 4 > Response 3 > Response 2 > Response 1.</p> <p>Explanation: Response 4 is the most accurate, concise, and directly mirrors the ground truth in both factual details and structure, making it the clearest and most reliable answer. Response 3 is similarly factually correct and well-structured but adds unnecessary embellishment ("puts even avocado-curious minds to shame") that, while engaging, introduces subjective commentary not present in the ground truth. Response 2 includes factual inaccuracies (mentions "irrigation pond, natural stone walkways, fountain, tailgating capability") that are not supported by the text, thereby reducing its reliability. Response 1 contains a clear contradiction—despite correctly listing most amenities, it confusingly states "the size of Oprah’s home is not given in the text," which is factually incorrect, making it the weakest response.</p>

208 D All Proof

209 Proof of Lemma 3.1

210 We prove by induction. First, for $t = 1$, we have

$$p_d(y_1 | x) = \frac{p_d(x, y_1)}{p_d(x)} \text{ and } p_{\theta^S}(y_1 | x) = \frac{p_{\theta^S}(x, y_1)}{p_d(x)}. \quad (7)$$

211 From Eq. (7), we reach the conclusion for $t = 1$. Assume that we have the conclusion for $i \leq t - 1$.

212 We further derive

$$\begin{aligned} p_d(x, y_{\leq t}) &= p_d(y_t | x, y_{< t}) p_d(x) \prod_{i=1}^{t-1} p_d(y_i | x, y_{< i}), \\ p_{\theta^S}(x, y_{\leq t}) &= p_{\theta^S}(y_t | x, y_{< t}) p_d(x) \prod_{i=1}^{t-1} p_{\theta^S}(y_i | x, y_{< i}). \end{aligned} \quad (8)$$

213 From Eq. (8), we reach the conclusion for t .

214 Proof of Theorem 3.3

215 Let $r_t^* \in \mathcal{R}_t^k$ be the solution of

$$\forall t : \max_{r_t \in \mathcal{R}_t^k} \mathbb{E}_{[x, y_{\leq t}] \sim D} \left[r_t([x, y_{< t}], y_t) - \mathbb{E}_{y'_t \sim \pi_{\theta^S}(\cdot | x, y_{< t})} [r_t([x, y_{< t}], y'_t)] \right]. \quad (9)$$

216 According to the consistency property, there exists $r^* \in \mathcal{R}^k$ such that $\forall t : r_t^*([x, y_{< t}], y_t) =$
 217 $r^*(x, y_{\leq t})$. We now prove that r^* is the solution of the following OP:

$$\max_{r \in \mathcal{R}^k} \mathbb{E}_{(x, y) \sim D, y' \sim \pi_{\theta^S}(\cdot | x)} \left[\sum_t \gamma^{t-1} r([x, y_{< t}], y_t) - \sum_t \gamma^{t-1} r([x, y_{< t}], y'_t) \right], \quad (10)$$

218 where we denote $r([x, y_{< t}], y_t) = r(x, y_{\leq t})$.

219 Let $\bar{r}^* \in \mathcal{R}^k$ be the solution of the OP in (10). We have

$$\begin{aligned} & \sum_t \gamma^{t-1} \bar{r}^*([x, y_{< t}], y_t) - \sum_t \gamma^{t-1} \bar{r}^*([x, y_{< t}], y'_t) = \sum_t \gamma^{t-1} [\bar{r}^*([x, y_{< t}], y_t) - \bar{r}^*([x, y_{< t}], y'_t)] \\ &= \sum_t \gamma^{t-1} [\bar{r}_t^*([x, y_{< t}], y_t) - \bar{r}_t^*([x, y_{< t}], y'_t)] \leq \sum_t \gamma^{t-1} [r_t^*([x, y_{< t}], y_t) - r_t^*([x, y_{< t}], y'_t)] \\ &= \sum_t \gamma^{t-1} [r^*([x, y_{< t}], y_t) - r^*([x, y_{< t}], y'_t)], \end{aligned} \quad (11)$$

220 where $\bar{r}_t^* \in \mathcal{R}_t^k$ are the consistent versions of $\bar{r}^* \in \mathcal{R}^k$ in the token-level families.

221 Finally, (11) indicates that r^* is the solution of the OP in (10).

222 Proof of Theorem 3.4

223 We consider

$$d_1(\mathbb{P}_d, \mathbb{P}_{\theta^S}) := \max_{r \in \mathcal{R}^k} \mathbb{E}_{(x, y) \sim D, y' \sim \pi_{\theta^S}(\cdot | x)} [R(x, y) - R(x, y')]. \quad (12)$$

224 Because if $r \in \mathcal{R}^k$ then $-r \in \mathcal{R}^k$, we have $d_1(\mathbb{P}_d, \mathbb{P}_{\theta^S}) \geq 0$ and $\min_{\theta^S \in \Theta} d_1(\mathbb{P}_d, \mathbb{P}_{\theta^S}) \geq 0$.
 225 Moreover, the minimization is obtained at 0 for θ^S such that $\mathbb{P}_{\theta^S} = \mathbb{P}_d$.

226 We now consider

$$d_2(\mathbb{P}_d, \mathbb{P}_{\theta^S}) := \max_{r \in \mathcal{R}^k} \mathbb{E}_{(x, y) \sim D, y' \sim \pi_{\theta^S}(\cdot | x)} \left[\sum_t \gamma^{t-1} r([x, y_{< t}], y_t) - \sum_t \gamma^{t-1} r([x, y_{< t}], y'_t) \right], \quad (13)$$

227 Because if $r \in \mathcal{R}^k$ then $-r \in \mathcal{R}^k$, we have $d_2(\mathbb{P}_d, \mathbb{P}_{\theta^S}) \geq 0$ and $\min_{\theta^S \in \Theta} d_2(\mathbb{P}_d, \mathbb{P}_{\theta^S}) \geq 0$.
 228 Moreover, the minimization is obtained at 0 for θ^S such that $\mathbb{P}_{\theta^S} = \mathbb{P}_d$ according to Lemma 1.

229 **Proof of Lemma 3.5**

230 Our proof is adopted from the proof of Lemma 4.2 in [6].

$$\begin{aligned}
& \max_{\pi_{\theta^S}} \mathbb{E}_{x, y_{<t} \sim D, y'_t \sim \pi_{\theta^S}(\cdot | x, y_{<t})} \left[\omega_t A_{\pi_{\theta_k^S}}([x, y_{<t}], y'_t) \right] - \beta D_{KL} \left(\pi_{\theta^S}(\cdot | [x, y_{<t}]) \parallel \pi_{\theta_k^S}(\cdot | [x, y_{<t}]) \right) \\
&= \max_{\pi_{\theta^S}} \mathbb{E}_{x, y_{<t} \sim D, y'_t \sim \pi_{\theta^S}(\cdot | x, y_{<t})} \left[\omega_t Q_{\pi_{\theta_k^S}}([x, y_{<t}], y'_t) - \omega_t V_{\pi_{\theta_k^S}}([x, y_{<t}]) - \beta \log \frac{\pi_{\theta^S}(y'_t | [x, y_{<t}])}{\pi_{\theta_k^S}(y'_t | [x, y_{<t}])} \right] \\
&= \max_{\pi_{\theta^S}} \mathbb{E}_{x, y_{<t} \sim D, y'_t \sim \pi_{\theta^S}(\cdot | x, y_{<t})} \left[\beta \log \frac{\pi_{\theta_k^S}(y'_t | [x, y_{<t}]) \exp \left\{ \frac{\omega_t}{\beta} Q_{\pi_{\theta_k^S}}([x, y_{<t}], y'_t) \right\}}{\pi_{\theta^S}(y'_t | [x, y_{<t}]) Z([x, y_{<t}]; \omega_t, \beta)} \right. \\
&\quad \left. - \omega_t V_{\pi_{\theta_k^S}}([x, y_{<t}]) + \log Z([x, y_{<t}]; \omega_t, \beta) \right] \\
&= \max_{\pi_{\theta^S}} \mathbb{E}_{x, y_{<t} \sim D, y'_t \sim \pi_{\theta^S}(\cdot | x, y_{<t})} \left[-\beta D_{KL} \left(\pi_{\theta^S}(y'_t | [x, y_{<t}]) \parallel \frac{\pi_{\theta_k^S}(y'_t | [x, y_{<t}]) \exp \left\{ \frac{\omega_t}{\beta} Q_{\pi_{\theta_k^S}}([x, y_{<t}], y'_t) \right\}}{Z([x, y_{<t}]; \omega_t, \beta)} \right) \right. \\
&\quad \left. - \omega_t V_{\pi_{\theta_k^S}}([x, y_{<t}]) + \log Z([x, y_{<t}]; \omega_t, \beta) \right].
\end{aligned}$$

231 This concludes our proof.

232 **Proof of Lemma 3.6**

233 Using the same derivations as in [6], we gain

$$\begin{aligned}
R(x, y) &= V_{\pi}([x, y^{<1}]) - \gamma^T V_{\pi}([x, y_{<T+1}]) \\
&\quad + \sum_{t=1}^T [\gamma^{t-1} (r([x, y_{<t}], y_t) + \gamma V_{\pi}([x, y_{<t+1}])) - V_{\pi}([x, y_{<t}])] \\
&= V_{\pi}([x, y^{<1}]) - V_{\pi}([x, y_{<T+1}]) \\
&\quad + \sum_{t=1}^T [r([x, y_{<t}], y_t) + V_{\pi}([x, y_{<t+1}]) - V_{\pi}([x, y_{<t}])],
\end{aligned}$$

234 since $\gamma = 1$.

235 We further have

$$\begin{aligned}
Q_{\pi}([x, y_{<t}], y_t) &= r([x, y_{<t}], y_t) + V_{\pi}([x, y_{<t+1}]), \\
A_{\pi}([x, y_{<t}], y_t) &= Q_{\pi}([x, y_{<t}], y_t) - V_{\pi}([x, y_{<t}]) \\
&= r([x, y_{<t}], y_t) + V_{\pi}([x, y_{<t+1}]) - V_{\pi}([x, y_{<t}]).
\end{aligned}$$

236 By noting that

$$V_{\pi}([x, y_{<T+1}]) = 0 \text{ and } V_{\pi}([x, y^{<1}]) = V_{\pi}([x, y'^{<1}]) = V_{\pi}([x, []]),$$

237 we gain

$$l(R(x, y) - R(x, y')) = l \left(\sum_t A_{\pi}([x, y_{<t}], y_t) - \sum_t A_{\pi}([x, y'_{<t}], y'_t) \right).$$

238 **Proof of Theorem 3.7**

239 We have

$$Q_{\pi_{\theta_k^S}}([x, y_{<t}], z) = \omega_t^{-1} \beta \log \frac{\pi_{\theta^S}^*(z | x, y_{<t})}{\pi_{\theta_k^S}(z | x, y_{<t})} + \omega_t^{-1} \beta \log Z([x, y_{<t}]; \omega_t, \beta). \quad (14)$$

240 We also have

$$l(R(x, y) - R(x, y')) = l \left(\sum_t \gamma^{t-1} A_{\pi_{\theta_k^S}}([x, y_{<t}], y_t) - \sum_t \gamma^{t-1} A_{\pi_{\theta_k^S}}([x, y'_{<t}], y'_t) \right). \quad (15)$$

241 We further derive as

$$\begin{aligned}
& \sum_t \gamma^{t-1} A_{\pi_{\theta_k^S}}([x, y_{<t}], y_t) = \sum_t \gamma^{t-1} [Q_{\pi_{\theta_k^S}}([x, y_{<t}], y_t) - V_{\pi_{\theta_k^S}}([x, y_{<t}])] \\
&= \sum_t \gamma^{t-1} [Q_{\pi_{\theta_k^S}}([x, y_{<t}], y_t) - \mathbb{E}_{y'_t \sim \pi_{\theta_k^S}} [Q_{\pi_{\theta_k^S}}([x, y_{<t}], y'_t)]] \\
&= \sum_t \gamma^{t-1} \left[\omega_t^{-1} \beta \log \frac{\pi_{\theta^S}^*(y_t | x, y_{<t})}{\pi_{\theta_k^S}(y_t | x, y_{<t})} + \omega_t^{-1} \beta \log Z([x, y_{<t}]; \omega_t, \beta) \right. \\
&\quad \left. \mathbb{E}_{y'_t \sim \pi_{\theta_k^S}} \left[\omega_t^{-1} \beta \log \frac{\pi_{\theta^S}^*(y'_t | x, y_{<t})}{\pi_{\theta_k^S}(y'_t | x, y_{<t})} + \omega_t^{-1} \beta \log Z([x, y_{<t}]; \omega_t, \beta) \right] \right] \\
&= \sum_t \gamma^{t-1} \left[\omega_t^{-1} \beta \log \frac{\pi_{\theta^S}^*(y_t | x, y_{<t})}{\pi_{\theta_k^S}(y_t | x, y_{<t})} - \mathbb{E}_{y'_t \sim \pi_{\theta_k^S}} \left[\omega_t^{-1} \beta \log \frac{\pi_{\theta^S}^*(y'_t | x, y_{<t})}{\pi_{\theta_k^S}(y'_t | x, y_{<t})} \right] \right] \\
&= \sum_t \gamma^{t-1} \left[\omega_t^{-1} \beta \log \frac{\pi_{\theta^S}^*(y_t | x, y_{<t})}{\pi_{\theta_k^S}(y_t | x, y_{<t})} + \omega_t^{-1} \beta D_{KL} \left(\pi_{\theta^S}^*(\cdot | x, y_{<t}) \| \pi_{\theta_k^S}(\cdot | x, y_{<t}) \right) \right] \\
&= \beta \sum_t \gamma^{t-1} \omega_t^{-1} \log \frac{\pi_{\theta^S}^*(y_t | x, y_{<t})}{\pi_{\theta_k^S}(y_t | x, y_{<t})} + \beta \sum_t \omega_t^{-1} \gamma^{t-1} D_{KL} \left(\pi_{\theta^S}^*(\cdot | x, y_{<t}) \| \pi_{\theta_k^S}(\cdot | x, y_{<t}) \right).
\end{aligned}$$

242 Finally, substituting to Eq. (15), we gain the conclusion.

References

- [1] Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36:53728–53741, 2023.
- [2] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [3] Ralph Allan Bradley and Milton E Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- [4] Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.
- [5] Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. Self-play fine-tuning converts weak language models to strong language models. *arXiv preprint arXiv:2401.01335*, 2024.
- [6] Yongcheng Zeng, Guoqing Liu, Weiyu Ma, Ning Yang, Haifeng Zhang, and Jun Wang. Token-level direct preference optimization. *arXiv preprint arXiv:2404.11999*, 2024.
- [7] Aiwei Liu, Haoping Bai, Zhiyun Lu, Yanchao Sun, Xiang Kong, Simon Wang, Jiulong Shan, Albin Madappally Jose, Xiaojiang Liu, Lijie Wen, et al. Tis-dpo: Token-level importance sampling for direct preference optimization with estimated weights. *arXiv preprint arXiv:2410.04350*, 2024.
- [8] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.
- [9] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- [10] Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro Von Werra, Cl  mentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.
- [11] AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, Ddl Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. Mistral 7b. arxiv 2023. *arXiv preprint arXiv:2310.06825*, 2024.
- [12] Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.
- [13] Edward Beeching, Cl  mentine Fourrier, Nathan Habib, Sheon Han, Nathan Lambert, Nazneen Rajani, Omar Sanseviero, Lewis Tunstall, and Thomas Wolf. Open llm leaderboard, 2023.
- [14] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac’h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024.
- [15] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.
- [16] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? *arXiv preprint arXiv:1905.07830*, 2019.

- [17] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.
- [18] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.
- [19] Stephanie Lin, Jacob Hilton, and Owain Evans. Truthfulqa: Measuring how models mimic human falsehoods. *arXiv preprint arXiv:2109.07958*, 2021.
- [20] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [21] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2023.
- [22] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arxiv* 2019. *arXiv preprint arXiv:1910.09700*, 2019.
- [23] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.
- [24] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.
- [25] Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.
- [26] Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. *arXiv preprint arXiv:2204.07705*, 2022.
- [27] Yulong Chen, Yang Liu, Liang Chen, and Yue Zhang. Dialogsum: A real-life scenario dialogue summarization dataset. *arXiv preprint arXiv:2105.06762*, 2021.
- [28] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.
- [29] Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. Dual-space knowledge distillation for large language models. *arXiv preprint arXiv:2406.17328*, 2024.
- [30] Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. Zero: Memory optimizations toward training trillion parameter models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pages 1–16. IEEE, 2020.
- [31] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.
- [32] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Lecture 6a overview of mini-batch gradient descent. *Coursera Lecture slides* <https://class.coursera.org/neuralnets-2012-001/lecture>, [Online], 2012.
- [33] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.
- [34] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

- 338 [35] Taku Kudo and John Richardson. Sentencepiece: A simple and language independent subword
339 tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*, 2018.
- 340 [36] Xinying Song, Alex Salcianu, Yang Song, Dave Dopson, and Denny Zhou. Fast wordpiece
341 tokenization. *arXiv preprint arXiv:2012.15524*, 2020.