AKD : Adversarial Knowledge Distillation For Large Language Models Alignment on Coding tasks

Ilyas Oulkadda [1]   Julien Perez [1]

Abstract

The widespread adoption of Large Language Models (LLMs) for code generation, exemplified by GitHub Copilot[1] surpassing a million users, highlights the transformative potential of these tools in improving developer productivity. However, this rapid growth also underscores critical concerns regarding the quality, safety, and reliability of the code they generate. As Code-LLMs evolve, they face significant challenges, including the diminishing returns of model scaling and the scarcity of new, high-quality training data. To address these issues, this paper introduces Adversarial Knowledge Distillation (AKD), a novel approach that leverages adversarially generated synthetic datasets to distill the capabilities of larger models into smaller, more efficient ones. By systematically stress-testing and refining the reasoning capabilities of Code-LLMs, AKD provides a framework for enhancing model robustness, reliability, and security while improving their parameter-efficiency. We believe this work represents a critical step toward ensuring dependable automated code generation within the constraints of existing data and the cost-efficiency of model execution.

1. Introduction

The rapid development and deployment of Large Language Models (LLMs), particularly those designed for code generation, such as GitHub Copilot, have started a transformative era in software development. These models hold the promise of significantly enhancing productivity by automating repetitive coding tasks and assisting developers with complex problem-solving. However, this evolution has also introduced pressing concerns regarding the quality, security, and ethical implications of the generated code. As developers increasingly depend on these models for critical applications, ensuring their outputs meet high standards of reliability and security has become imperative.

Traditional methods for aligning LLMs rely heavily on

---

[1]A coding extension powered by a Code-LLM to assist in code completion tasks

human-annotated datasets, which provide direct feedback and corrections. While effective, this approach faces challenges in scalability and adaptability due to the growing size of modern LLMs and the rapidly evolving nature of coding practices. The need for more sustainable and efficient alignment methodologies has become apparent, prompting research into alternative strategies.

On the other hand, generating datasets synthetically using LLMs can be beneficial in terms of scalability and efficiency (Gunasekar et al., 2023b; Long et al., 2024). However, it doesn't guarantee that the synthetic data will address all relevant cases. Synthetic data, generated in an open-ended manner, might lack the diversity and complexity of real-world scenarios, leading to gaps in the model's understanding and performance. This can result in models that perform well on synthetic benchmarks but struggle with real-world applications. Therefore, it's crucial to combine synthetic data generation with rigorous validation against real-world data to ensure the model's robustness and applicability.

To address these challenges, we propose Adversarial Knowledge Distillation (AKD), a novel framework designed to align Code-LLMs with functional and performance standards without the extensive reliance on human annotation. AKD integrates Direct Preference Optimization (DPO) (Rafailov et al., 2023) into a curriculum-driven adversarial dataset generation process (Gunasekar et al., 2023a; OpenAI et al., 2021). This approach enables distillation of knowledge (Hinton et al., 2015) from a larger, more capable "teacher" model into a smaller, computationally efficient "student" model through an automatically defined curriculum by leveraging adversarially generated synthetic datasets of code exercices.

Concretely, AKD begins with the teacher model generating coding tasks, complete with function descriptions and solution outlines, which serve as a curated curriculum for the student model. Adversarial interactions are then used to stress-test the student model's reasoning and alignment capabilities, systematically exposing weaknesses and guiding improvements. The optimization process relies on DPO loss applied to the adversarial curriculum, allowing the student model to iteratively refine its alignment with the teacher's distribution. Unlike conventional methods, such as Self-
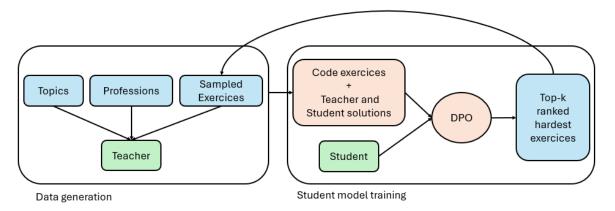
Figure 1. Adversarial Knowledge Distillation framework comprises three components, differentiated by color: models (green), training processes (red), and synthetic data generation elements (blue). In the initial step, topics and profession seeds are used to generate an initial dataset. Profession is acting as a persona to help diversify the dataset. For subsequent iterations, exercises are sampled based on margin rewards calculated from the previous training cycle. Once the exercises are selected or generated, the teacher model produces the "chosen" solutions, while the student model generates the "rejected" solutions. The resulting dataset, containing both chosen and rejected solutions, is then used to train the student model via Direct Preference Optimization (DPO). Margin rewards, which reflect the difference in performance between the teacher and student on specific tasks, are calculated to rank exercises. This ranking informs the sampling process for generating the next dataset, ensuring a targeted and iterative improvement cycle for the student model.

Supervised Fine-Tuning (SFT) based on next-token prediction, or DPO applied to unordered datasets, our approach leverages curriculum learning (Sukhbaatar et al., 2018) to organize training data into a sequence that maximizes learning efficacy.

Preliminary results demonstrate the effectiveness of AKD in improving model performance and robustness. On the HumanEval (Chen et al., 2021) benchmarks, models trained with our adversarial curriculum exhibit higher reliability and security in generated code compared to those trained using traditional alignment methods. Furthermore, training logs reveal significant gains in reward margins, underscoring the effectiveness of adversarial datasets in enhancing the student model's reasoning and adaptability.

This work seeks to advance the state of the art in aligning LLMs for automated code generation by addressing the critical challenges of quality, scalability, and ethical alignment and we aim to offer a scalable and sustainable pathway for developing smaller, more efficient models while ensuring their outputs remain secure, reliable, and aligned with industry standards. In the following sections, we provide a review of related work in Section 2, introduce our adversarial adaptations of DPO in Section 3, and analyze the performance of our approach in Section 4.

2. Related Work

Data Annotation: Data annotation is still foundational for training and refining aligned Large Language Models (LLMs). Traditional methods often rely on manual annotation, which, while effective, faces significant scalability challenges as models and datasets grow in size and complexity. Manual annotation involves human annotators labeling data points, which can be time-consuming and costly, especially for large datasets. As an example, (Li et al., 2023) employed 1,399 crowdworkers from 35 countries to annotate a dataset for identifying Personally Identifiable Information (PII) in source code. This highlights the extensive human effort required and the inherent difficulties in maintaining consistency and scalability across such diverse and large-scale annotation efforts. The process involved coordinating a large number of annotators, ensuring they understood the annotation guidelines, and managing the quality of the annotations. The challenges included variations in annotation quality due to differences in annotator expertise and cultural backgrounds, as well as the need for continuous monitoring and feedback to maintain consistency. As LLMs require vast amounts of annotated data to achieve high performance, the traditional manual annotation approach becomes challenging. The need for efficient and scalable data production methods has led to the exploration of alternative approaches, such as semi-supervised learning, active learning, and automated annotation tools (Ruder & Plank, 2018; Xie et al., 2020; Chen et al., 2020). These methods aim to reduce the reliance on manual annotation by leveraging machine learning techniques to assist in the annotation process, thereby improving scalability and efficiency.

Self-Instruct Approaches: As a consequence to the expensive cost of annotation, leveraging larger models to autonomously generate high-quality, well-structured data has emerged as a scalable alternative to manual annotation. Notable examples include the phi-1.5 model (Gunasekar et al.,

2023b), a 1.3B parameters model trained on high-quality, generated textbooks, and the *Cosmopedia* dataset (Eldan & Li, 2023), which further demonstrates the utility of advanced models for data generation. The phi-1.5 model has been trained on synthetic "textbook-like" data, achieving performance comparable to models five times its size. The *Cosmopedia* dataset, generated by the Mixtral-8x7B-Instruct-v0.1 model (Jiang et al., 2024a), contains over 30 million files and 25B tokens, making it the largest open synthetic dataset to date. Moreover, incorporating topic sampling techniques enhances dataset diversity and ensures comprehensive coverage of scenarios, which is particularly critical for generating robust and adaptable models.

Adversarial and Curriculum Learning: Regarding automatic curriculum and dataset generation, adversarial learning has shown promise in various domains such as board games, video games, and robotics. This approach leverages the competitive nature of adversarial setups to generate increasingly complex task curricula, fostering progressive learning. One notable technique is "asymmetric self-play" (OpenAI et al., 2021), where two agents, called Alice and Bob, engage in a competitive game. Alice is tasked with proposing challenging goals, while Bob aims to solve them. This dynamic leads to the discovery of highly diverse and complex goals without any human priors, thereby enhancing the learning process. For instance, in robotic manipulation tasks, asymmetric self-play has enabled the development of goal-conditioned policies that can generalize to previously unseen goals and objects (OpenAI et al., 2021). This method has been particularly effective in zero-shot generalization, where the learned policy can adapt to new tasks without additional training. Another key technique is "intrinsic motivation and automatic curricula" (Sukhbaatar et al., 2018). This approach pits two versions of the same agent against each other, where one agent proposes tasks and the other attempts to complete them. Through an appropriate reward structure, the agents automatically generate a curriculum of exploration, enabling unsupervised training. This method has been successfully applied in environments that can be reset or are nearly reversible, allowing the agent to learn about its environment in an unsupervised manner. The generator network proposes tasks specified as goal states, and the agent learns to perform a wide set of tasks efficiently. These insights motivates our adversarial training approach, where dynamically generated data progressively challenges the model. By mirroring a curriculum-learning framework tailored to Code-LLMs, we aim to enhance the model's ability to handle complex coding tasks. The adversarial setup ensures that the model is continually presented with new and challenging scenarios, fostering robust and adaptable learning. This approach can improves the model's performance on known tasks but also equips it with the ability to generalize to new, unseen tasks, making it more versatile

and effective in real-world applications.

Direct Preference Optimization and Knowledge Distillation: Direct Preference Optimization (DPO) (Rafailov et al., 2023) represents a significant advancement in the alignment process of Large Language Models (LLMs). Unlike Proximal Policy Optimization (PPO), which rely on a reward model and can suffer from instability, DPO transforms the optimization challenge into a classification loss problem. This approach directly compares preferred and non-preferred outputs, streamlining the training process and reducing computational complexity. By doing so, DPO enhances also stability and makes the alignment of LLM outputs with human or teacher-generated preferences more efficient and effective. As the size of modern models continues to grow, concerns about their deployability for local use, especially for LLMs, have become more pronounced. Knowledge distillation (Hinton et al., 2015) offers a compelling solution to this challenge. This technique enables smaller models to learn from the output (soft labels) of larger teacher models, effectively transferring knowledge while reducing the overall model size. By combining soft labels with hard-labeled annotated data during training, knowledge distillation ensures that the smaller model retains the performance benefits of the larger model without the computational overhead. Our approach builds on the concept of knowledge distillation but adapts it specifically for adversarial fine-tuning scenarios. Instead of relying on hard labels, we focus on dynamic, adversarially generated datasets to guide the alignment of the student model. This adaptation not only simplifies the fine-tuning process but also ensures that the student model is exposed to a diverse and challenging set of examples, enhancing its robustness and adaptability. Knowledge distillation has been successfully applied in various domains, including natural language processing and computer vision, where it has helped create more efficient and deployable models. By leveraging these techniques, we aim to develop LLMs that are both high-performing and practical for real-world applications.

3. Methodology

Adversarial Knowledge Distillation (AKD) introduces adversarial dynamics into the Direct Preference Optimization (DPO) algorithm. This is achieved by orchestrating an adversarial game between two models: a Teacher and a Student. The Teacher, being a larger and more proficient language model in the domain of coding, is tasked with generating coding exercises (prompts) and providing solutions that are assumed to align with expert human standards. These solutions are labeled as "chosen" answers. Conversely, the Student, a smaller model, attempts to solve the same exercises, and its solutions are labeled as "rejected" due to their presumed lesser alignment with expert preferences.

Our DPO setup involves a dataset with three key components: the prompt p (coding exercise), the chosen solution c (provided by the Teacher), and the rejected solution r (generated by the Student). This structured format facilitates the execution of DPO training iterations. To introduce adversariality, we conduct multiple rounds of these training iterations conditioned by the former errors of the Student model. At the end of each iteration, we calculate the loss for each training sample, enabling us to determine the rewards for each instance.

The loss function for a given sample can be defined as:

$$L(x, c, r) = -\log \left(\frac{\exp(R(c, x))}{\exp(R(c, x)) + \exp(R(r, x))} \right)$$

where $R(c)$ and $R(r)$ are the rewards for the chosen and rejected solutions, respectively.

Adversarial Step

The core of the adversariality in our framework lies in how we identify and target the weaknesses of the Student model. A critical metric in this setup is the *margin reward*, which measures the difference between the rewards assigned to the chosen and rejected answers. The margin reward M is computed as:

$$M = R(c) - R(r)$$

where $R(c)$ and $R(r)$ represent the rewards for the chosen and rejected answers, respectively.

A small margin indicates that the Student model struggles to differentiate between the correct and incorrect solutions. This forms the foundation of the adversariality: we aim to focus on these challenging areas to push the model's boundaries.

To operationalize this, we apply a softmax over the negative margins, which transforms the margin rewards into a probability distribution. Prompts with lower margins (indicating greater difficulty) are assigned higher probabilities, ensuring that these weaknesses are emphasized in subsequent training. The probability $P(p)$ of selecting a prompt p is defined as:

$$P(p) = \frac{\exp(-M(p))}{\sum_{p' \in B} \exp(-M(p'))}$$

where B represents the batch of prompts.

This adversarial sampling process occurs at the batch level, rather than at the level of individual samples. This approach not only diversifies the topics and challenges presented to the Student model but also ensures that the dataset adapts dynamically to target its weaknesses. The iterative nature of this process drives the adversariality by continually generating and focusing on data that challenges the model, fostering robust learning.

The final step involves using these selectively sampled exercises to generate new prompts targeting the identified weaknesses of the Student model. Initial attempts to generate similar exercises directly from the hardest samples resulted in adversarial datasets that were smaller and overly similar to the initial dataset. To address this issue, we implemented three distinct prompting strategies to diversify and enrich the generated adversarial exercises:

Incremental Approach: This strategy gradually increases the difficulty of the exercises, ensuring that the Student model is continually challenged but not overwhelmed. By starting with the hardest samples identified in the previous iteration and incrementally increasing the difficulty, we aim to create a smooth learning curve. The difficulty level D of a new exercise is given by:

$$D_{\text{new}} = D_{\text{hard}} + \Delta D$$

where D_{hard} is the difficulty of the hardest sample and ΔD is a small increment. This incremental approach helps the Student model to build on its existing knowledge and skills, progressively enhancing its capabilities. The small increments ensure that the model is exposed to new challenges that are just beyond its current abilities, promoting effective learning and adaptation.

Opposite Approach: This approach is motivated by the need to challenge the Student model's assumptions and biases. By generating exercises that appear similar to familiar tasks but introduce unexpected variations, we force the model to adapt to new and unpredictable scenarios. The similarity S between the new exercise and the original exercise is maintained within a certain threshold:

$$S(p_{\text{new}}, p_{\text{original}}) \leq \theta$$

where θ is the similarity threshold. This strategy ensures that the model does not rely on superficial patterns or shortcuts but instead develops a deeper understanding of the underlying principles. By confronting the model with exercises that defy its expectations, this approach encourages it to generalize better and become more robust to diverse and unseen situations.

Deceptive Approach: This strategy focuses on creating exercises that appear deceptively simple but conceal underlying complexity. The goal is to challenge the Student model's ability to discern and handle hidden intricacies. The complexity C of the exercise is hidden behind a deceptive simplicity:

$$C(p_{\text{deceptive}}) > C(p_{\text{simple}})$$

where p_{simple} is a straightforward exercise. This approach is particularly effective in training the model to avoid overconfidence and to develop a more nuanced understanding of the tasks. By presenting exercises that require deeper analysis

and careful consideration, we promote the model's ability to handle subtle and complex problems, enhancing its overall problem-solving skills.

These diversified strategies ensure that the adversarial datasets are not only richer in variety but also more robust in challenging the Student model across different dimensions. The resulting iterative process dynamically adapts to the evolving capabilities of the model. As the Student model improves, the Teacher model continually generates new challenges that push the boundaries of the Student's current abilities. This adaptive mechanism ensures that the training remains consistently robust and diverse, preventing the model from plateauing or overfitting to specific types of exercises. By continually introducing new and varied challenges, we foster a learning environment that promotes sustained growth and improvement. All training details, including the exact prompts used for dataset generation, the parameters for each prompting strategy, and the iterative training schedule, are provided in the Appendix.

4. Experiments

In this section, we address research questions related to the performance and effectiveness of our proposed method. Traditionally, knowledge distillation combines soft labels, i.e. probability distributions produced by the teacher model, and hard labels, i.e. ground truth labels, to train a student model from scratch. However, due to computational constraints, our experiments focus on fine-tuning pre-trained models rather than training from scratch. In our approach, the teacher's soft labels are treated as hard labels for fine-tuning. Despite this limitation, we hypothesize that our method remains effective as a fine-tuning strategy, offering performance gains through adversarially ordered training steps.

We evaluate our approach using two widely-adopted code generation benchmarks: HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021). To ensure rigorous assessment of functional correctness, we employ the EvalPlus framework (Liu et al., 2023), evaluating on both the original benchmarks and their enhanced EvalPlus+ variants. These augmented versions retain the original problem statements while substantially expanding the test suites—HumanEval+ adds an average of 80× more tests per problem compared to HumanEval, and MBPP+ provides 35× more tests than MBPP (Liu et al., 2023). This comprehensive evaluation strategy enables us to measure both base performance and robustness against subtle implementation errors.

Our distillation framework is evaluated across contemporary open-source models spanning 1B to 8B parameters, including Meta's general-purpose Llama-3.1-8B and Llama-3.2-1B (Grattafiori et al., 2024), DeepSeek's reasoning-optimized R1 series (DS-R1-Qwen-7B and DS-R1-Llama-

Table 1. Models used in our distillation experiments.

Model	Context Length	Size
Llama-3.1-8B	128K	8B
Llama-3.2-1B	128K	1B
Phi-1.5	4500	1.3B
DeepSeek-R1-Qwen-7B	4500	7B
DeepSeek-R1-Llama-8B	130K	8B
Qwen2.5-7B	4500	7B

8B) (DeepSeek-AI et al., 2025), Qwen's code-specialized Qwen2.5-Coder-7B (Hui et al., 2024), and Microsoft's compact Phi-1.5. This selection encompasses both general instruction-following architectures and domain-specific variants, with more details provided in Table 1. The spectrum of model sizes and architectural specializations enables comprehensive analysis of knowledge transfer efficacy across different capability profiles.

The rest of section is organized into three subsections. First, we evaluate whether our method outperforms self-supervised fine-tuning approaches. Second, we assess the contribution of adversarial training steps to model performance. Finally, we explore the applicability of our teacher-student framework for speculative decoding (Leviathan et al., 2023), focusing on its impact on speed and accuracy in text generation. These experiments aim to validate the capabilities of AKD across multiple dimensions, demonstrating its effectiveness in improving fine-tuning, enhancing alignment, and supporting efficient decoding strategies.

Since we were limited in resources, we were not able to run AKD at larger scale generating larger datasaets. Our experiments are done on rather small datasets varying from 700k tokens to 3M tokens using Llama 3 tokenizer. We believe that the results could be scaled up with dataset size as seen with the DS-R1 and Llama 3.2 1B pair showing higher performance with a dataset size of 3M tokens. However, we also notice high improvements on the Qwen2.5-Coder 7B and Phi 1.5 pair with only 700k tokens.

4.1. Experimental settings

For data generation, detailed in Table 3, the teacher model uses a longer maximum sequence length only for exercise creation, while solutions use the same length as the student model to avoid bias and ensure fair output sizes. To optimize memory efficiency during training, we employ gradient accumulation of 4 and gradient checkpointing, with a low learning rate reflecting a later training stage post-pretraining, instruction fine-tuning, and alignment. The framework performs best initially but degrades with more repetitions, so we limit repetitions. Future work could explore this with larger datasets and more complex tasks. We use LoRA (Hu et al., 2021) for fine-tuning, targeting attention projection matrices and MLP layers to enhance efficiency and perfor-

Teacher	Student	Size	HumanEval		HumanEval+		MBPP		MBPP+	
		(M)	T	S (Δ)	T	S (Δ)	T	S (Δ)	T	S (Δ)
DeepSeek-R1-Qwen-7B	Llama-3.2-1B	3.0	39.6	36.5 (**+3.6**)	37.8	29.2 (0.0)	43.1	41.5 (**+8.7**)	37.3	34.9 (**+6.9**)
Llama-3.1-8B	Phi-1.5	6.4	69.5	39.0 (+3.1)	62.8	31.0 (+1.0)	68.3	53.9 (-1.1)	54.8	43.9 (-0.8)
Qwen2.5-Coder-7B	Phi-1.5	1.1	61.6	38.2 (+2.3)	53.0	31.7 (+1.7)	76.9	52.9 (-2.1)	62.9	43.3 (-1.1)
Qwen2.5-Coder-7B	Llama-3.2-1B	0.7	61.6	34.7 (+1.8)	53.0	32.3 (**+3.2**)	76.9	32.5 (-0.3)	62.9	28.0 (0.0)
Llama3.1-8B	Llama-3.2-1B	1.7	69.5	30.4 (-2.5)	62.8	28.0 (-1.2)	68.3	34.7 (+1.9)	54.8	29.9 (+1.9)

Table 2. Knowledge Distillation Performance. Size column indicates millions of tokens in the training dataset. T: Teacher score, S: Student score with improvement delta (Δ) from initial version. The data we generate is mainly in the HumanEval format however a small portion is reserved for natural language instruction similar to MBPP format. Constant improvements are seen on HumanEval except for the Llama pair, where the 1B model is already distilled from larger counterparts.
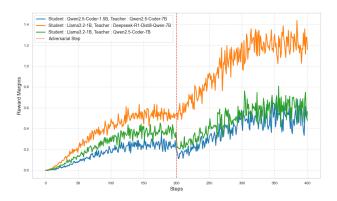

Figure 2. Margin Rewards during training: At the 200-step mark, an adversarial step generates a new dataset, continuing training with the same optimizer states. A small performance dip is observed in some experiments due to increased exercise difficulty.

mance, the hyperparameters are detailed in Table 4.

4.2. Performance Comparison: Proposed Framework vs. Self-Supervised Fine-Tuning

We compare the performance of AKD against traditional self-supervised fine-tuning approaches. Self-supervised methods rely on the model's ability to learn from the data itself without external supervision. While these methods have shown promise, they often struggle with capturing the nuanced preferences and complexities that a teacher model can provide. By leveraging the teacher's soft labels as hard labels, AKD aims to bridge this gap, offering a more guided and effective fine-tuning process. We evaluate the performance of both methods on a variety of benchmarks to determine if AKD provides a significant advantage.

Concretely, we evaluate the performance of Adversarial Knowledge Distillation (AKD) against a baseline self-

Parameter	Value
Number of subtopics per topic	10
Number of exercises per subtopic	10
Oracle temperature	1
Oracle max length	2048
Student temperature	0.6
Student max length	512
Number of steps	150
Per device train batch size	4
Learning rate	5e-6
Beta	0.01
Repetitions	5
Quantization	BF16
Gradient accumulation steps	4
Learning rate scheduler type	cosine
Optimizer	Paged AdamW

Table 3. Main Hyperparameters for Model Training and Data Generation.

LoRA Configuration	Value
Rank (r)	16
LoRA alpha	32
LoRA dropout	0.05
Bias	none
Target modules	{q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj}

Table 4. Configuration Parameters for LoRA adapters.

supervised fine-tuning (SFT) method using next-token prediction loss. Both methods are benchmarked on the HumanEval dataset, with AKD demonstrating unique advantages. The AKD framework creates a preference dataset by generating coding questions and solutions using a teacher

model. These questions are then attempted by the student model, and the resulting answers are paired to form a preference dataset, where the teacher's response is marked as the preferred solution. The training process uses a contrastive loss function to align the student's predictions closer to the teacher's distribution, maximizing learning efficiency.

For the baseline, we explore two approaches to self-supervised fine-tuning. The first approach uses the same synthetic dataset generated by the teacher for AKD, where the questions and teacher solutions are concatenated to create a training corpus. The second approach fine-tunes the student model on the APPS dataset, a challenging coding benchmark comprising approximately 5,000 samples. While both approaches yield equivalent performance on HumanEval, significant insights emerge from the dataset comparison.

Table 5 shows that AKD achieves a comparable 38% accuracy on the HumanEval benchmark, matching the performance of SFT on the APPS dataset. Notably, the AKD-generated dataset, consisting of only 1.6k samples, achieves the same results as the handcrafted APPS dataset, underscoring the quality and efficiency of AKD's dataset generation process. These findings highlight that AKD not only matches the performance of traditional self-supervised fine-tuning on a significantly smaller dataset but also offers a scalable and automated approach to creating high-quality datasets for coding tasks. Future work will explore scaling AKD to additional domains and larger datasets while maintaining its competitive performance.

4.3. Benefits of Adversarial Training Steps

Here, we evaluate the specific contributions of adversarial training steps to the overall performance of the model. Adversarial training introduces a dynamic element to the learning process, continually challenging the model with increasingly difficult tasks. This approach not only helps in identifying and addressing the model's weaknesses but also promotes robustness and adaptability. We analyze the performance metrics at different stages of the adversarial training process to quantify the benefits of this approach. By comparing models fine-tuned with and without adversarial steps, we aim to demonstrate the tangible improvements brought by our method. Unlike standard training, where the entire dataset is presented in random order, our method introduces an iterative approach. Exercises are sequentially presented with increasing difficulty, leveraging a curriculum designed to maximize the student model's learning efficiency.

To assess the impact of this strategy, we compare our framework to a baseline DPO approach. In the DPO baseline, the entire synthetic dataset is concatenated and presented as a single batch during training. By contrast, our adversarial approach incrementally adapts the dataset, using performance feedback from the student model to rank and order the ex-

ercises. This ranking ensures that the student encounters progressively more challenging tasks, simulating a teacher-like guidance mechanism.

As shown in Table 6, our adversarial framework consistently outperforms the DPO baseline, achieving a 4 percentage point improvement in accuracy. This result highlights the importance of gradual difficulty progression, which appears to enhance the student model's learning dynamics and overall performance. Furthermore, this finding aligns with prior research on curriculum learning, suggesting that structured training paradigms can yield significant gains in model efficacy.

4.4. Speculative Decoding with Teacher-Student Framework

Finally, we explore the potential of our teacher-student framework in the context of speculative decoding. Speculative decoding is a technique aimed at accelerating text generation by predicting future tokens based on current predictions. This method relies heavily on the accuracy and speed of the model's predictions. We investigate how our AKD framework can enhance speculative decoding by providing more accurate and efficient predictions. By evaluating the speed and accuracy of text generation with and without our framework, we assess the practical benefits of integrating AKD with speculative decoding techniques.

In this subsection, we explore the applicability of our framework for speculative decoding, leveraging the teacher as the main model and the student as the assistant. Speculative decoding is a decoding strategy that accelerates text generation by delegating initial token predictions to a smaller assistant model and verifying these predictions with the main model. The primary objective is to benchmark the accuracy of this setup relative to the number of forward passes required on the main model, thereby evaluating the trade-off between generation speed and performance.

To achieve effective speculative decoding, the assistant model must satisfy two key criteria. Firstly, it should be smaller in size to ensure computational efficiency. Secondly, and most importantly, the output token distribution of the assistant must closely resemble that of the main model. A higher degree of similarity minimizes the number of forward passes to the main model, as fewer corrections are required for the assistant's predictions.

While speculative decoding is traditionally implemented using distilled or quantized versions of the main model, these approaches present challenges: distillation of an entire model remains computationally expensive, and quantization introduces complexities in ensuring compatibility and maintaining performance. Our adversarial knowledge distillation (AKD) framework was proposed as an alternative to generate assistant models tailored for speculative decoding by

Method	Accuracy (HumanEval, %)	Dataset Size (Samples)
Self-Supervised Fine-Tuning (AKD dataset)	38	1.6k
Self-Supervised Fine-Tuning (APPS train)	38	5k
Adversarial Knowledge Distillation	38	1.6k

Table 5. Comparison of Benchmark Performance Between Adversarial Knowledge Distillation and Self-Supervised Fine-Tuning. Evaluated on Qwen2.5 Coder 7B / Llama 3.2 1B

Method	Accuracy (HumanEval, %)	Improvement (%)
DPO Baseline	35	–
Adversarial Training	**38**	+3

Table 6. Performance Comparison: Adversarial Training vs. DPO Baseline

fine-tuning the student model on datasets generated by the teacher.

However, our experiments did not demonstrate improvements in speculative decoding performance using teacher-student pairs trained within the same model family (e.g., Llama 8B as the teacher and Llama 1B as the student). Speculative decoding requires both the assistant and main models to have compatible tokenizers, which restricted our experiments to models from the same family. This limitation likely contributed to the lack of improvement, as models within the same family are typically trained on overlapping datasets. Consequently, the teacher's generated text closely resembles the pretraining data, leaving limited room for the student to learn beyond the shared knowledge already captured during pretraining.

We theorize that this limitation is due to the inherent saturation of knowledge within a single model family. Since both teacher and student models are trained on similar corpora, the teacher's prompts and solutions do not introduce novel challenges for the student. This lack of novelty restricts the student's ability to fine-tune its token distribution to better align with the teacher, which is critical for successful speculative decoding.

While our results did not confirm the efficacy of AKD for speculative decoding in this constrained setup, we hypothesize that using models from different families with distinct training data but same tokenizers may yield better results. Future work should explore cross-family speculative decoding setups, where the teacher and student are chosen from different architectures, enabling richer adversarial interactions and more diverse datasets for fine-tuning the student.

5. Conclusion

In this paper, we have introduce Adversarial Knowledge Distillation, a novel alignment framework for language models, with a focus on code-focused LLMs. AKD demonstrates promise as a flexible, task-agnostic method adaptable to a variety of domains. Our experiments revealed that AKD can yield meaningful performance improvements with limited data, outperforming traditional methods such as standard fine-tuning and Direct Preference Optimization in scenarios where pre-existing datasets are unavailable. Through the use of synthetic datasets comprising approximately 1400 samples, we observed that AKD enables student models to improve relative to their teachers in code-related tasks. This underscores its potential as an efficient approach to aligning models without the need for large curated datasets. Scaling the dataset size and diversifying its themes represent promising directions to further enhance AKD's effectiveness. In the future, methods like AKD could be scaled and adapted to other well-defined tasks, leveraging its task-agnostic nature. This flexibility makes it applicable beyond code, paving the way for broader utility across domains. For code-specific applications, additional avenues of improvement could be explored by building on the current work. One promising direction is integrating compiler feedback to refine the quality of synthetic datasets. Compiler feedback could serve as a valuable signal to ensure datasets align closely with task requirements, enhancing model performance for coding challenges.

References

AI@Meta. Llama 3 model card. 2024. URL https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., and Sutton, C. Program synthesis with large language models, 2021. URL https://arxiv.org/abs/2108.0 7732.

Ben Allal, L., Muennighoff, N., Kumar Umapathi, L., Lipkin, B., and von Werra, L. A framework for the evaluation of code generation models. https://github.com /bigcode-project/bigcode-evaluation-h arness, 2022.

Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. Cosmopedia, 2024. URL https://huggin gface.co/datasets/HuggingFaceTB/cosm opedia.

Bowman, S. R., Hyun, J., Perez, E., Chen, E., Pettit, C., Heiner, S., Lukošiūtė, K., Askell, A., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., Olah, C., Amodei, D., Amodei, D., Drain, D., Li, D., Tran-Johnson, E., Kernion, J., Kerr, J., Mueller, J., Ladish, J., Landau, J., Ndousse, K., Lovitt, L., Elhage, N., Schiefer, N., Joseph, N., Mercado, N., DasSarma, N., Larson, R., McCandlish, S., Kundu, S., Johnston, S., Kravec, S., Showk, S. E., Fort, S., Telleen-Lawton, T., Brown, T., Henighan, T., Hume, T., Bai, Y., Hatfield-Dodds, Z., Mann, B., and Kaplan, J. Measuring progress on scalable oversight for large language models, 2022.

Chen, J., Yang, Z., and Yang, D. Mixtext: Linguistically-informed interpolation of hidden space for semi-supervised text classification. *arXiv preprint arXiv:2004.12239*, 2020.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025. URL https://arxiv.org/abs/2501.12948.

Eldan, R. and Li, Y. Tinystories: How small can language models be and still speak coherent english?, 2023.

Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Vaughan, A., Yang, A., Fan, A., Goyal, A., Hartshorn, A., Yang, A., Mitra, A., Sravankumar, A., Korenev, A., Hinsvark, A., Rao, A., Zhang, A., Rodriguez, A., Gregerson, A., Spataru, A., Roziere, B., Biron, B., Tang, B., Chern, B., Caucheteux, C., Nayak, C., Bi, C., Marra, C., McConnell, C., Keller, C., Touret, C., Wu, C., Wong, C., Ferrer, C. C., Nikolaidis, C., Allonsius, D., Song, D., Pintz, D., Livshits, D., Wyatt, D., Esiobu, D., Choudhary, D., Mahajan, D., Garcia-Olano, D., Perino, D., Hupkes, D., Lakomkin, E., AlBadawy, E., Lobanova, E., Dinan, E., Smith, E. M., Radenovic, F., Guzmán, F., Zhang, F., Synnaeve, G., Lee, G., Anderson, G. L., Thattai, G., Nail, G., Mialon, G., Pang, G., Cucurell, G., Nguyen, H., Korevaar, H., Xu, H., Touvron, H., Zarov, I., Ibarra, I. A., Kloumann, I., Misra, I., Evtimov, I., Zhang, J., Copet, J., Lee, J., Geffert, J., Vranes, J., Park, J., Mahadeokar, J., Shah, J., van der Linde, J., Billock, J., Hong, J., Lee, J., Fu, J., Chi, J., Huang, J., Liu, J., Wang, J., Yu, J., Bitton,

J., Spisak, J., Park, J., Rocca, J., Johnstun, J., Saxe, J., Jia, J., Alwala, K. V., Prasad, K., Upasani, K., Plawiak, K., Li, K., Heafield, K., Stone, K., El-Arini, K., Iyer, K., Malik, K., Chiu, K., Bhalla, K., Lakhotia, K., Rantala-Yeary, L., van der Maaten, L., Chen, L., Tan, L., Jenkins, L., Martin, L., Madaan, L., Malo, L., Blecher, L., Landzaat, L., de Oliveira, L., Muzzi, M., Pasupuleti, M., Singh, M., Paluri, M., Kardas, M., Tsimpoukelli, M., Oldham, M., Rita, M., Pavlova, M., Kambadur, M., Lewis, M., Si, M., Singh, M. K., Hassan, M., Goyal, N., Torabi, N., Bashlykov, N., Bogoychev, N., Chatterji, N., Zhang, N., Duchenne, O., Çelebi, O., Alrassy, P., Zhang, P., Li, P., Vasic, P., Weng, P., Bhargava, P., Dubal, P., Krishnan, P., Koura, P. S., Xu, P., He, Q., Dong, Q., Srinivasan, R., Ganapathy, R., Calderer, R., Cabral, R. S., Stojnic, R., Raileanu, R., Maheswari, R., Girdhar, R., Patel, R., Sauvestre, R., Polidoro, R., Sumbaly, R., Taylor, R., Silva, R., Hou, R., Wang, R., Hosseini, S., Chennabasappa, S., Singh, S., Bell, S., Kim, S. S., Edunov, S., Nie, S., Narang, S., Raparthy, S., Shen, S., Wan, S., Bhosale, S., Zhang, S., Vandenhende, S., Batra, S., Whitman, S., Sootla, S., Collot, S., Gururangan, S., Borodinsky, S., Herman, T., Fowler, T., Sheasha, T., Georgiou, T., Scialom, T., Speck-bacher, T., Mihaylov, T., Xiao, T., Karn, U., Goswami, V., Gupta, V., Ramanathan, V., Kerkez, V., Gonguet, V., Do, V., Vogeti, V., Albiero, V., Petrovic, V., Chu, W., Xiong, W., Fu, W., Meers, W., Martinet, X., Wang, X., Wang, X., Tan, X. E., Xia, X., Xie, X., Jia, X., Wang, X., Gold-schlag, Y., Gaur, Y., Babaei, Y., Wen, Y., Song, Y., Zhang, Y., Li, Y., Mao, Y., Coudert, Z. D., Yan, Z., Chen, Z., Papakipos, Z., Singh, A., Srivastava, A., Jain, A., Kelsey, A., Shajnfeld, A., Gangidi, A., Victoria, A., Goldstand, A., Menon, A., Sharma, A., Boesenberg, A., Baevski, A., Feinstein, A., Kallet, A., Sangani, A., Teo, A., Yunus, A., Lupu, A., Alvarado, A., Caples, A., Gu, A., Ho, A., Poul-ton, A., Ryan, A., Ramchandani, A., Dong, A., Franco, A., Goyal, A., Saraf, A., Chowdhury, A., Gabriel, A., Bharambe, A., Eisenman, A., Yazdan, A., James, B., Maurer, B., Leonhardi, B., Huang, B., Loyd, B., Paola, B. D., Paranjape, B., Liu, B., Wu, B., Ni, B., Hancock, B., Wasti, B., Spence, B., Stojkovic, B., Gamido, B., Montalvo, B., Parker, C., Burton, C., Mejia, C., Liu, C., Wang, C., Kim, C., Zhou, C., Hu, C., Chu, C.-H., Cai, C., Tindal, C., Feichtenhofer, C., Gao, C., Civin, D., Beaty, D., Kreymer, D., Li, D., Adkins, D., Xu, D., Testuggine, D., David, D., Parikh, D., Liskovich, D., Foss, D., Wang, D., Le, D., Holland, D., Dowling, E., Jamil, E., Mont-gomery, E., Presani, E., Hahn, E., Wood, E., Le, E.-T., Brinkman, E., Arcaute, E., Dunbar, E., Smothers, E., Sun, F., Kreuk, F., Tian, F., Kokkinos, F., Ozgenel, F., Cag-gioni, F., Kanayet, F., Seide, F., Florez, G. M., Schwarz, G., Badeer, G., Swee, G., Halpern, G., Herman, G., Sizov, G., Guangyi, Zhang, Lakshminarayanan, G., Inan, H., Shojanazeri, H., Zou, H., Wang, H., Zha, H., Habeeb, H., Rudolph, H., Suk, H., Aspegren, H., Goldman, H., Zhan, H., Damlaj, I., Molybog, I., Tufanov, I., Leontiadis, I., Veliche, I.-E., Gat, I., Weissman, J., Geboski, J., Kohli, J., Lam, J., Asher, J., Gaya, J.-B., Marcus, J., Tang, J., Chan, J., Zhen, J., Reizenstein, J., Teboul, J., Zhong, J., Jin, J., Yang, J., Cummings, J., Carvill, J., Shepard, J., McPhie, J., Torres, J., Ginsburg, J., Wang, J., Wu, K., U, K. H., Saxena, K., Khandelwal, K., Zand, K., Matosich, K., Veeraraghavan, K., Michelena, K., Li, K., Jagadeesh, K., Huang, K., Chawla, K., Huang, K., Chen, L., Garg, L., A, L., Silva, L., Bell, L., Zhang, L., Guo, L., Yu, L., Moshkovich, L., Wehrstedt, L., Khabsa, M., Avalani, M., Bhatt, M., Mankus, M., Hasson, M., Lennie, M., Reso, M., Groshev, M., Naumov, M., Lathi, M., Keneally, M., Liu, M., Seltzer, M. L., Valko, M., Restrepo, M., Patel, M., Vyatskov, M., Samvelyan, M., Clark, M., Macey, M., Wang, M., Hermoso, M. J., Metanat, M., Rastegari, M., Bansal, M., Santhanam, N., Parks, N., White, N., Bawa, N., Singhal, N., Egebo, N., Usunier, N., Mehta, N., Laptev, N. P., Dong, N., Cheng, N., Chernoguz, O., Hart, O., Salpekar, O., Kalinli, O., Kent, P., Parekh, P., Saab, P., Balaji, P., Rittner, P., Bontrager, P., Roux, P., Dollar, P., Zvyagina, P., Ratanchandani, P., Yuvraj, P., Liang, Q., Alao, R., Rodriguez, R., Ayub, R., Murthy, R., Nayani, R., Mitra, R., Parthasarathy, R., Li, R., Hogan, R., Battey, R., Wang, R., Howes, R., Rinott, R., Mehta, S., Siby, S., Bondu, S. J., Datta, S., Chugh, S., Hunt, S., Dhillon, S., Sidorov, S., Pan, S., Mahajan, S., Verma, S., Yamamoto, S., Ramaswamy, S., Lindsay, S., Lindsay, S., Feng, S., Lin, S., Zha, S. C., Patil, S., Shankar, S., Zhang, S., Zhang, S., Wang, S., Agarwal, S., Sajuyigbe, S., Chintala, S., Max, S., Chen, S., Kehoe, S., Satter-field, S., Govindaprasad, S., Gupta, S., Deng, S., Cho, S., Virk, S., Subramanian, S., Choudhury, S., Goldman, S., Remez, T., Glaser, T., Best, T., Koehler, T., Robinson, T., Li, T., Zhang, T., Matthews, T., Chou, T., Shaked, T., Vontimitta, V., Ajayi, V., Montanez, V., Mohan, V., Kumar, V. S., Mangla, V., Ionescu, V., Poenaru, V., Mi-hailescu, V. T., Ivanov, V., Li, W., Wang, W., Jiang, W., Bouaziz, W., Constable, W., Tang, X., Wu, X., Wang, X., Wu, X., Gao, X., Kleinman, Y., Chen, Y., Hu, Y., Jia, Y., Qi, Y., Li, Y., Zhang, Y., Zhang, Y., Adi, Y., Nam, Y., Yu, Wang, Zhao, Y., Hao, Y., Qian, Y., Li, Y., He, Y., Rait, Z., DeVito, Z., Rosnbrick, Z., Wen, Z., Yang, Z., Zhao, Z., and Ma, Z. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C. C. T., Giorno, A. D., Gopi, S., Javaheripi, M., Kauffmann, P., de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Behl, H. S., Wang, X., Bubeck, S., Eldan, R., Kalai, A. T., Lee, Y. T., and Li, Y. Textbooks are all you need, 2023a.

Gunasekar, S., Zhang, Y., Aneja, J., Mendes, C. C. T., Giorno, A. D., Gopi, S., Javaheripi, M., Kauffmann, P.,

de Rosa, G., Saarikivi, O., Salim, A., Shah, S., Behl, H. S., Wang, X., Bubeck, S., Eldan, R., Kalai, A. T., Lee, Y. T., and Li, Y.-F. Textbooks are all you need. *ArXiv*, abs/2306.11644, 2023b. URL https://api.sema nticscholar.org/CorpusID:259203998.

Guo, D., Zhu, Q., Yang, D., Xie, Z., Dong, K., Zhang, W., Chen, G., Bi, X., Wu, Y., Li, Y. K., Luo, F., Xiong, Y., and Liang, W. Deepseek-coder: When the large language model meets programming – the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.1 4196.

Hinton, G., Vinyals, O., and Dean, J. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for nlp, 2019.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models, 2021.

Hui, B., Yang, J., Cui, Z., Yang, J., Liu, D., Zhang, L., Liu, T., Zhang, J., Yu, B., Lu, K., Dang, K., Fan, Y., Zhang, Y., Yang, A., Men, R., Huang, F., Zheng, B., Miao, Y., Quan, S., Feng, Y., Ren, X., Ren, X., Zhou, J., and Lin, J. Qwen2.5-coder technical report, 2024. URL https://arxiv.org/abs/2409.12186.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de Las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts. *ArXiv*, abs/2401.04088, 2024a. URL https://api.semanticscholar.org/CorpusID:266844877.

Jiang, A. Q., Sablayrolles, A., Roux, A., Mensch, A., Savary, B., Bamford, C., Chaplot, D. S., de las Casas, D., Hanna, E. B., Bressand, F., Lengyel, G., Bour, G., Lample, G., Lavaud, L. R., Saulnier, L., Lachaux, M.-A., Stock, P., Subramanian, S., Yang, S., Antoniak, S., Scao, T. L., Gervet, T., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mixtral of experts, 2024b.

Le, H., Wang, Y., Gotmare, A. D., Savarese, S., and Hoi, S. C. H. Coderl: Mastering code generation through pretrained models and deep reinforcement learning, 2022. Advances in Neural Information Processing Systems 35.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Li, R., Allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., Liu, Q., Zheltonozhskii, E., Zhuo, T. Y., Wang, T., Dehaene, O., Davaadorj, M., Lamy-Poirier, J., Monteiro, J., Shliazhko, O., Gontier, N., Meade, N., Zebaze, A., Yee, M.-H., Uma-pathi, L. K., Zhu, J., Lipkin, B., Oblokulov, M., Wang, Z., Murthy, R., Stillerman, J., Patel, S. S., Abulkhanov, D., Zocca, M., Dey, M., Zhang, Z., Fahmy, N., Bhattacharyya, U., Yu, W., Singh, S., Luccioni, S., Villegas, P., Kunakov, M., Zhdanov, F., Romero, M., Lee, T., Timor, N., Ding, J., Schlesinger, C., Schoelkopf, H., Ebert, J., Dao, T., Mishra, M., Gu, A., Robinson, J., Anderson, C. J., Dolan-Gavitt, B., Contractor, D., Reddy, S., Fried, D., Bahdanau, D., Jernite, Y., Ferrandis, C. M., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. Starcoder: may the source be with you!, 2023.

Liu, J., Xia, C. S., Wang, Y., and Zhang, L. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net /forum?id=1qvx610Cu7.

Liu, J., Xie, S., Wang, J., Wei, Y., Ding, Y., and Zhang, L. Evaluating language models for efficient code generation. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IBCB MeAhmC.

Long, L., Wang, R., Xiao, R., Zhao, J., Ding, X., Chen, G., and Wang, H. On llms-driven synthetic data generation, curation, and evaluation: A survey. In *Annual Meeting of the Association for Computational Linguistics*, 2024. URL https://api.semanticscholar.org/ CorpusID:270688337.

OpenAI, O., Plappert, M., Sampedro, R., Xu, T., Akkaya, I., Kosaraju, V., Welinder, P., D'Sa, R., Petron, A., d. O. Pinto, H. P., Paino, A., Noh, H., Weng, L., Yuan, Q., Chu, C., and Zaremba, W. Asymmetric self-play for automatic goal discovery in robotic manipulation, 2021.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022.

Rafailov, R., Sharma, A., Mitchell, E., Ermon, S., Manning, C. D., and Finn, C. Direct preference optimization: Your language model is secretly a reward model, 2023.

Rozière, B., Gehring, J., Gloeckle, F., Sootla, S., Gat, I., Tan, X. E., Adi, Y., Liu, J., Sauvestre, R., Remez, T., Rapin, J., Kozhevnikov, A., Evtimov, I., Bitton, J., Bhatt, M., Ferrer, C. C., Grattafiori, A., Xiong, W., Défossez, A., Copet, J., Azhar, F., Touvron, H., Martin, L., Usunier, N., Scialom, T., and Synnaeve, G. Code llama: Open foundation models for code, 2024.

Ruder, S. and Plank, B. Semi-supervised learning for natural language processing. *arXiv preprint arXiv:1804.09530*, 2018.

Sanh, V., Debut, L., Chaumond, J., and Wolf, T. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter, 2020. URL https://arxiv.org/abs/19 10.01108.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. Trust region policy optimization, 2017a. URL https://arxiv.org/abs/1502.05477.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017b.

Sukhbaatar, S., Lin, Z., Kostrikov, I., Synnaeve, G., Szlam, A., and Fergus, R. Intrinsic motivation and automatic curricula via asymmetric self-play, 2018. ICLR 2018.

Team, Q. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/b log/qwen2.5/.

Wu, F., Liu, X., and Xiao, C. Deceptprompt: Exploiting llm-driven code generation via adversarial natural language instructions, 2023.

Xie, Q., Dai, Z., Hovy, E., Luong, M.-T., and Le, Q. V. Unsupervised data augmentation for consistency training. *arXiv preprint arXiv:1904.12848*, 2020.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. A comprehensive survey on transfer learning, 2020. URL https://arxiv.org/abs/ 1911.02685.

6. Appendix

This appendix provides detailed prompt templates for generating synthetic Python programming exercises. These prompts are designed to create diverse and challenging exercises for learners, ranging from fundamental to advanced levels. Each category serves a specific purpose in dataset generation, ensuring a mix of conceptual understanding and practical application.

6.1. Topics Generation

This prompt assists in generating structured subtopics for a given Python concept. The goal is to ensure a logical progression of topics, covering both fundamental and advanced aspects. The generated subtopics should be distinct, build upon each other, and allow for the creation of multiple types of exercises, including theory, practice, and debugging.

```
As a Python textbook author, create {
    n} distinct subtopics for the
    main topic '{topic}'.

Requirements:
- Each subtopic should be broad
    enough to generate multiple
    exercise types (theory, practice,
     debugging).
- Include both fundamental and
    advanced concepts.
- Ensure topics build on each other
    in a logical learning progression
    .
- Mix conceptual and practical
    application areas.
- Avoid overlapping subtopics.

Format your response as a Python list
     of strings, like this:
['Basic String Operations', 'String
    Formatting Methods', 'Regular
    Expressions']

Return only the Python list with no
    additional text or explanation.
```

6.2. Initial Synthetic Dataset Generation, Code Completion Version

This prompt generates Python code completion exercises tailored to a specific topic. Each exercise is framed within a real-world professional context to enhance engagement and relevance. The structure ensures that learners focus on writing functional code while adhering to given constraints.

```
Create {n} unique and challenging
    Python code completion exercises
    on the topic of "{topic}".
Each exercise should be framed in the
    context of a {profession}'s work
    to make it more engaging.

Structure each exercise as follows:

```
def function_name(parameters):
    """
    Description of the task or
        problem to solve, providing
        all necessary context.
    """
    # Solution code starts here (in
        Python)
```

Guidelines:
- Avoid using classes
- Make exercises challenging
```

6.3. Initial Synthetic Dataset Generation, Natural Language Version

This prompt generates Python programming exercises described in natural language. The exercises are embedded within realistic professional scenarios, ensuring domain relevance. Each exercise includes problem statements, input-output specifications, test cases, and constraints to guide learners.

```
Create {n} unique and challenging
    Python programming exercises on
    the topic of "{topic}",
framed within the context of a {
    profession}'s daily work
    scenarios.

Structure each exercise as follows:

[PROBLEM]
Problem: [Natural language
    description that presents the
    programming challenge in a {
    profession}-related scenario]

Input: [Clearly specify the input
    format using domain-specific
    examples]

Output: [Clearly specify the expected
    output format]
```

```
Examples:
[Provide 2-3 test cases with
    profession-relevant data and
    explanations]

Constraints:
[State any constraints or special
    considerations]
[PROBLEM]

Example scenario structure:
- For a chef: "Given a list of recipe
    preparation times, find the
    optimal cooking schedule..."
- For an architect: "Given dimensions
    of building materials, calculate
    the most efficient arrangement
    ..."

Guidelines:
- Use profession-specific terminology
    and realistic scenarios
- Make the problem mathematically
    sound while maintaining
    professional context
- Include domain-relevant example
    data in test cases
- Do NOT provide the solution, only
    the problem description
- Make SURE to write the problem
    between the [PROBLEM] tokens
```

6.4. Adversarial Dataset Generation, Incremental Approach

This prompt generates exercises that incrementally increase complexity based on a given reference problem. These exercises introduce additional edge cases and nuanced difficulties, encouraging deeper problem-solving skills.

```
Based on the reference exercise '{
    reference}', generate {n} new
    coding exercises that introduce
    additional edge cases and
    increased complexity.

Requirements:
- Build upon the original task,
    adding nuanced complexity.
- Avoid using classes, but require
    complex logic and multiple edge
    cases.
- Provide a solution following the
    exercise.

Format:
```
```

```
def function_name(parameters):
    \"\"\"Exercise description with
        additional complexity.\"\"\"
    # Solution code here
```

6.5. Adversarial Dataset Generation, Opposite Approach

This prompt creates exercises that challenge the assumptions of a given reference problem. The aim is to encourage adaptability in solving unexpected variations of a familiar topic.

```
Create {n} adversarial coding
    exercises for the topic related
    to '{reference}' that challenge
    conventional assumptions made in
    the original exercise.

Requirements:
- Shift expected assumptions,
    requiring the model to adapt to
    unexpected scenarios.
- Keep exercises challenging and
    avoid using classes.
- Provide solutions immediately after
     each exercise.

Format:
```
def function_name(parameters):
    """Exercise with altered
        assumptions."""
    # Solution code here
```
```

6.6. Adversarial Dataset Generation, Deceptive Complexity Approach

This prompt generates exercises that appear simple at first glance but contain hidden complexities. These problems test the learner's ability to identify and handle underlying difficulties in problem-solving.

```
Generate {n} coding exercises
    inspired by '{reference}' that
    appear simple but involve hidden
    complexities.

Requirements:
- Exercises should look
    straightforward but require
    complex solutions.
- Avoid classes and focus on
    deceptive problem setups.
- Solutions should immediately follow
     the exercise.

Format:
```
def function_name(parameters):
    """Exercise with hidden
        complexities.""
    # Solution code here
```
```