NDP: Next Distribution Prediction as a More Broad Target

Anonymous ACL submission

Abstract

Large language models (LLMs) trained on next-token prediction (NTP) paradigm have demonstrated powerful capabilities. However, the existing NTP paradigm contains several limitations, particularly related to planned task complications and error propagation during inference. In our work, we extend the critique of NTP, highlighting its limitation also due to training with a narrow objective: the prediction of a sub-optimal one-hot distribution. Based on this insight, we introduce Next Distribution Prediction (NDP), which uses statistical distributions to replace the one-hot targets, enhancing learning without extra online training time. We conducted experiments across translation, general task, language transfer, and medical domain adaptation. Compared to NTP, NDP can achieve up to +2.97 COMET improvement in translation tasks, +0.61 average improvement in general tasks, and incredible +10.75 average improvement in the vertical domain (e.g., Medical Domain). This demonstrates the concrete benefits of addressing the target narrowing problem, pointing to a new direction for future work on improving NTP.

1 Introduction

Large Language Models (LLMs) are predominantly trained using NTP paradigm. However, this approach has been subject to criticism, primarily focusing on two key issues: (1) the inability to perform tasks requiring advanced planning, such as look-ahead tasks (Kambhampati et al., 2024b; Bachmann and Nagarajan, 2024), and (2) error propagation during inference. These critiques have prompted various improvements, including methods to incorporate planning for future tokens during training or inference (Kambhampati et al., 2024a; Monea et al., 2023; Chen et al., 2023; Gloeckle et al., 2024; Cai et al., 2024).

We argue that the NTP paradigm is constrained not only by its short-term focus in the temporal dimension but also by its restrictive candidate selection process. Specifically, during training, the model is conditioned to treat the next token for a given prefix as the sole correct target, effectively striving to approximate a one-hot distribution. This scenario is analogous to a student being "tormented" by a capricious teacher who insists on learning distinct and unique correct answers at each step, as illustrated in Figure 1. Such a rigid "tormenting" learning process fails to fully leverage the extensive learning capabilities of large models, which are inherently capable of exploring multiple solution pathways simultaneously.

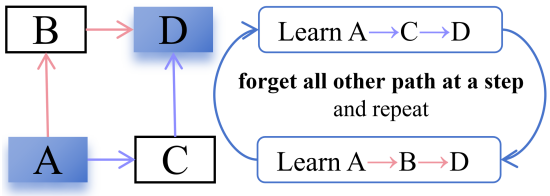

Figure 1: The "torture" of NTP's learning dilemma. Instead of learning two paths at the same time, they repeatedly learn one path and forget the other paths. Although this forgetting and learning objective may not be fully achieved due to the nature of stochastic gradient descent, this tendency also hinders the model's learning.

This observation raises an question: How can we identify and utilize all possible paths for a model to address a given problem effectively? Drawing inspiration from Huh et al. (2024), who proposed that a model's ultimate representation should function as a statistical model of the underlying reality, we propose that incorporating statistical methods can lead to a more comprehensive learning objective within the training dataset. By doing so, we aim to overcome the limitations of the NTP paradigm, enabling models to harness their full potential in learning multiple pathways to solutions.

To this end, we introduce Next Distribution Prediction (NDP), a method that improves the training objective of LLMs with the help of statistical mod-

els. This method analyzes the training corpus to identify the same prefixes and their corresponding successor tokens, and then converts the frequency of the successor tokens into a distribution, replacing the original one-hot distribution of the NTP. NDP can also leverage unsupervised data to further enhance the distribution, thereby achieving similar results without the need for continued pretraining on domain-specific data. This provides a potential solution for unifying continued pretraining and instruction fine-tuning. In the analysis section, we provide two perspectives: by comparing the similarities in the distributions of NTP, NDP, and LLM, as well as the convergence endpoints of NTP and NDP, we further demonstrate the advantages of NDP over NTP as a training paradigm.

Our extensive experiments across various models, tasks, and evaluation metrics demonstrate significant performance improvements. Moreover, NDP enables the simultaneous use of supervised and unsupervised data for training, effectively allowing for continued pre-training during fine-tuning. This feature is particularly advantageous for domain adaptation and language transfer scenarios. NDP outperforms NTP, showing improvements of up to 2.97 COMET points in translation tasks, an average gain of 0.61 points in general tasks, and a remarkable average increase of 10.75 points in the medical domain.

2 Related Work

2.1 Calibration During Training

Our work shares similarities with output probability calibration methods, as both aim to mitigate overconfidence and align output probabilities with true probabilities. Prominent calibration techniques during training include loss function modification (Ren et al., 2024; Li et al., 2020; Lin et al., 2018), label smoothing (Liang et al., 2024; Wei et al., 2022; Malagutti et al., 2024), Noise Injection(Sam and Kolter, 2023; Gao et al., 2019)

Research on loss function modification often attributes the discrepancy between predicted and real-world probabilities to maximum likelihood estimation. This has led to efforts to replace cross-entropy (e.g., negative log-likelihood) with alternative loss functions, introducing significant computational overhead and sensitive parameters. In contrast, NDP can be easily integrated into existing training frameworks without incurring additional training costs, yielding substantial improvements.

While NDP supports smoothing, its primary advantage stems from addressing the issue of narrow candidates rather than smoothing per se. NDP guarantees a non-one-hot distribution, allowing for multi-discrete value distributions rather than only continuous ones. Given the expanding vocabulary sizes in modern language models, the correct next token candidates cannot span the entire vocabulary range. For large language models requiring high-precision alignment, introducing noise across the entire vocabulary can result in downstream task performance inferior to that achieved with one-hot distributions from NTP.

2.2 Improvement on Next Token Prediction

Earlier criticisms of the NTP training paradigm were all focused on the time dimension, which led to many improvements. Monea et al. (2023) was inspired by Speculative Sampling (Chen et al., 2023), using the LLMs itself as a draft model, thus allowing the LLMs to output multiple tokens at once during the inference stage, implicitly achieving long-term planning and alleviating the short-term issues to some extent. Gloeckle et al. (2024) achieved consistent improvements in efficiency and performance on code tasks by training shared model backbones and multiple independent output heads and adopting speculative decoding with Medusa-like tree attention (Cai et al., 2024) during inference, indicating that this training paradigm has advantages in large-scale models.

These studies are completely orthogonal to our perspective. We primarily focus on the issues brought by narrow candidates, with the hope of jointly optimizing the NTP process.

2.3 Knowledge Distillation

We can further evaluate the effectiveness of NDP compared to NTP from the perspective of knowledge distillation. NDP can be seen as token-level, while NTP is sentence-level (Kim and Rush, 2016). This perspective illuminates NTP's limitations. Yuan et al. (2023) demonstrated that in knowledge distillation, student models more readily assimilate soft labels compared to one-hot labels. Wei et al. (2024) observed that the efficacy of sentence-level versus token-level distillation correlates with student model size, with larger models benefiting more from token-level approaches. Empirically, most research utilizing black-box Large Language Models (LLMs), such as instruction data synthesis (Xu et al., 2023), employs sentence-level

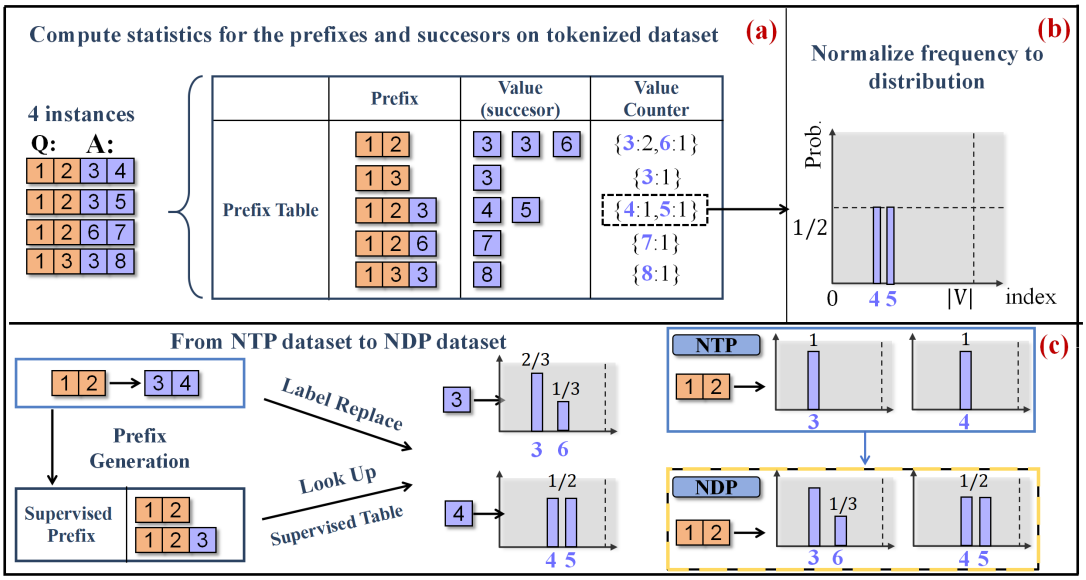

Figure 2: Overall framework of simplified NDP. The numbers in the squares represent token. (a) Count the successor words of the same prefix string in the training dataset to form a prefix table. (b) Convert each counter in the prefix table into a probability distribution through normalization. (c) Replace the labels in the original dataset using the probability distribution. Through these three steps, we convert the one-hot distribution NTP dataset into a statistical distribution NDP dataset.

distillation. While effective, sentence-level distillation alone has not enabled open-source LLMs to match the performance of GPT-4-turbo/GPT-4 (OpenAI et al., 2024) . Conversely, Gemma2-9B (Team et al., 2024) achieved performance comparable to LLaMA3-8B (Dubey et al., 2024) with only 9T pretraining tokens, attributable to its use of token-level distillation. These findings support NDP's superior performance over NTP.

It should be noted that there is an essential difference between dataset-based knowledge distillation (NTP/NDP) and model-based knowledge distillation. First, we cannot bypass the dataset-based distillation paradigm to obtain a pre-trained model; therefore, it can be said that model-based distillation must be built upon the dataset-based distillation paradigm. Model-based KD cannot replace NTP, but NDP can. Second, KD-based training methods require cooperating NTP to achieve good results such as Hybrid Distillation (Hinton et al., 2015; Romero et al., 2015), which also incurs significant training overhead. Typically, KD requires an additional teacher model, which has parameters that are more than ten times larger than the student model, leading to substantial memory usage and increased training time. In contrast, NDP does not need to be combined with NTP and does not incur additional training overhead.

3 Next Distribution Prediction Paradigm

In this section, we will provide a detailed description of how our method, NDP, incorporates the aforementioned statistical distribution concept into the actual model training process. Meanwhile, we provide a brief explanation in the Appendix B on how the NTP paradigm processes training data into distributions.

Almost all datasets can be categorized as either unsupervised or supervised datasets. Let's take supervised datasets as an example, since self-supervised datasets can be regarded as a special case of supervised datasets where the instruction/input is empty. This process can be divided into three sub-processes: First, learn the prefix table through statistical analysis of the dataset (Figure 2(a)). Second, convert value counter from prefix table to distribution (Figure 2(b)). And third, replace the training targets in the original dataset from one-hot distributions to non-one-hot distributions (Figure 2(c)). We will elaborate the details of each sub-process in subsequent sections. We also compared the storage/computation time overhead of NDP and NTP in Appendix C.

3.1 Learning Prefix Tables

The specific process is illustrated in Figure 2(a). Given a sentence, we use all its prefix sequences

as keys and the corresponding successor tokens as values to form several key-value pairs. Across the entire training set, the key-value pairs formed by different sentences are merged based on the identical key, and corresponding values will collectively form a frequency Counter.

It is evident that the one-hot distribution derived from NTP is a specialized form of a prefix table. When keys do not overlap in the table generated from the entire dataset, the supervised distribution becomes identical to the NTP distribution. In practice, we separately compile two prefix tables from the starting position of the question and the answer part, respectively referred to as the supervised table and the causal language modeling (CLM) table. In this way, we can separately extract supervised information and pretraining information from the dataset, making it easier for us to handle the two distributions more effectively in the following sections. If we want to use NDP to replace the NTP pretraining, we only need to compute the CLM table.

3.2 Converting Distributions from Prefix Table

Figure 2(b) shows the process of converting each element in the value counter into a distribution on the model vocabulary dimension. For each counter, we create a tensor with a dimension of the vocabulary size, extracting the indices and corresponding counts from the frequency counter and setting the tensor accordingly. Then, we convert this tensor into a probability distribution via L1 norm or softmax. In our preliminary analysis, using the softmax method yielded better results compared to the L1 norm method. Therefore, we opted for the softmax method in our experiments. Instead of applying the softmax function directly to the entire frequency vector, we applied it to the counted parts and then placed the transformed values back into their original index within the distribution. This approach prevents the softmax from producing a uniform distribution over the large vocabulary vector.

This process can form a non-one-hot distribution, but it remains sparse. We also provide a novel frequency to probability method in Appendix D, which efficiently converts the frequency vector into a distribution while controlling the amount of introduced noise (i.e., the probability values assigned to parts that were originally zero).

3.3 Replacing Origin One-hot Target

After properly handling the token-level distribution, we can simply traverse the original dataset to replace the training targets. In Figure 2(c), we provide an example with a sentence. First, we decompose a sentence into corresponding keys as in Figure 2(a). Then, we use the keys to look up the corresponding table and obtain the distribution that we transformed in Figure 2(b).

We employ a simple linear weighted fusion as Equation 1.

$$D_{mix} = \alpha D_{supervised} + (1 - \alpha) D_{CLM} \quad (1)$$

where α is a hyperparameter constrained to the interval [0, 1]. We substitute the original one-hot label target with D_{mix}. Through ablation analysis, we found that the best performance is achieved when alpha is set to 0.8. This indicates that during the instruction fine-tuning phase, the model is primarily learning the mapping from problems to answers in the problem space. It is important to note that when encountering a blank distribution in the fusion objects, we do not perform fusion but instead retain the original distribution. In other words, we consistently assign zero weight to blank distributions during the fusion process.

At this point, we have completed the data processing part. The next step in the training process is the regular teacher-forcing as NTP. NDP has another very interesting use: we can use a large amount of unlabeled text to further enhance the CLM table. This process essentially unifies pretraining and fine-tuning. We will demonstrate this later in Section 4.3 .

4 Experiments

4.1 General Tasks for Large Language Models

In this subsection, we aim to explore the impact of using NDP for instruction fine-tuning (IFT) on the base model for general tasks. The specific experimental setup is as follows.

Model & Baseline We conducted experiments on Gemma-2B (Team et al., 2024) and LLaMA3-8B (Dubey et al., 2024). We used LoRA (Hu et al., 2022) to train LLaMA3-8B. We mainly compare NDP with NTP, label smoothing.

Dataset We selected a mixture of Alpaca-GPT4 (Peng et al., 2023), Math (Hendrycks et al., 2021b),

	GSM	MMLU	HE	TruQA	BBH	ARC-C	TriQA	AE	SCIQ	WG	IFeval	Avg.
Gemma	19.56	42.12	23.78	33.05	35.94	40.36	47.38	27.20	94.30	65.67	14.42	40.34
+NTP	**24.49**	40.85	31.71	41.59	37.40	44.54	**42.73**	27.54	92.30	**66.61**	19.41	42.65
+LS	22.37	40.90	30.49	40.29	35.80	44.11	40.52	27.27	**93.30**	66.46	13.12	41.33
+NDP	23.81	**41.34**	**35.98**	**42.84**	**37.77**	**44.88**	42.57	**28.65**	91.90	66.38	**19.78**	**43.26**
LLaMA3	54.44	65.57	37.20	43.91	62.52	50.43	71.21	33.70	96.30	78.22	10.17	54.88
+NTP	**53.30**	62.37	37.80	44.04	60.13	**51.28**	**63.64**	33.38	**96.60**	77.82	14.97	54.12
+LS	50.80	58.23	35.37	43.73	51.14	51.02	57.69	**33.51**	96.50	**78.22**	17.19	52.13
+NDP	53.15	**62.99**	**39.02**	**44.05**	**61.86**	51.11	62.70	33.24	96.50	77.98	**17.93**	**54.59**

Table 1: Evaluation results on general tasks. The benchmark abbreviations in the table: GSM (GSM8k), MMLU (Massive Multitask Language Understanding), HE (HumanEval), TruQA (TruthfulQA), BBH (Big-Bench Hard), ARC-C (ARC-Challenge), TriQA (TriviaQA), AE (AgiEval), SCIQ (SCIQ), WG (WinoGrande), and IFeval (IFeval). LS means Label Smoothing here.

and Code (Zheng et al., 2024) as the instruction fine-tuning (IFT) dataset. This combination is similar to the typical mix of general text, code, and math used in pretraining, with a total dataset size of 220K instances. The evaluation comprised 11 benchmarks that broadly cover the model's general reasoning, knowledge Q&A, math, coding, fact, and instruction following capabilities. More detailed benchmark information can be found in Appendix E.

Evaluation framework Our evaluation process primarily leveraged the lm-evaluation-harness framework (Gao et al., 2023), with the exception of coding tasks, for which we utilized the evaluation scripts from the OpenAI/HumanEval repository. The evaluation setting closely follow those outlined in the LLaMA3 evaluation protocol[1].

Results The experimental results are summarized in Table 1. From the result, we observe some phenomena:

- **NDP has more advantages in complex tasks.**. We observed that NDP consistently outperformed NTP on MMLU, HumanEval, TruthfulQA, BBH, and IFeval. These benchmarks focus on reasoning, coding, factual accuracy, and instruction-following capabilities. Meanwhile, NTP exhibited advantages on GSM8K and TriviaQA, suggesting that multiple solution pathways training might not be essential for elementary math tasks and knowledge-based Q&A, as answers are often direct and reasoning paths are relatively straightforward. This observation can be categorized as NDP demonstrating more significant advantages in complex tasks.

- **Failure of Label Smoothing**. Contrary to expectations, the label smoothing technique did not yield performance gains. In fact, it underperforms relative to the NTP method. We hypothesize that the meaningless noise during the instruction fine-tuning phase may have degraded the quality of the fine-tuning data. This observation indicates that the critical importance of data quality over quantity in the instruction fine-tuning process.

4.2 Translation Task for Encoder-Decoder Models

In this subsection, we aim to answer the following questions: (1) Does our method work effectively for models with smaller parameter sizes? (2) Can our method benefit specific downstream tasks? Although we have demonstrated that NDP can benefit general, broad tasks, further discussion on adaptation to specific task can still be argued.

Model & Baseline The T5 model (Raffel et al., 2023) is an excellent choice because we will select a 400M decoder-only LLaMA in the latter experiment. Using T5 would allow us to observe the impact on encoder-decoder models as well. We selected three sizes of the T5 1.1 version models: small (77M), base (248M), and large (783M). Here we only compare with NTP, since in preliminary experiments, label smoothing has already shown a similar drop in performance as general tasks.

Dataset & Metric We selected 200k bilingual sentence pairs in the en-de direction from IWSLT17

[1] LLaMA3 evaluation protocol

		IWSLT17		WMT22		Avg.		Avg. Δ	
		BLEU	COMET	BLEU	COMET	BLEU	COMET	BLEU	COMET
T5_small	NTP	11.51	**55.49**	7.39	**48.43**	9.45	**51.96**	**+0.30**	-0.18
	NDP	**11.56**	55.39	**7.93**	48.16	**9.75**	51.78		
T5_base	NTP	19.97	68.54	**15.48**	61.90	17.73	65.22	**+0.91**	**+1.97**
	NDP	**21.87**	**70.66**	15.41	**63.72**	**18.64**	**67.19**		
T5_large	NTP	23.63	76.42	17.49	71.26	20.56	73.84	**+1.96**	**+1.17**
	NDP	**25.70**	**77.29**	**19.34**	**72.72**	**22.52**	**75.01**		

Table 2: T5 series evaluated on IWSLT17 & WMT22 with BLEU and COMET22.

	MedQA	MedMCQA	PubMedQA	CareQA	Avg.	Avg. Δ	MMLU	FLOPS
Qwen2	44.46	46.57	47.30	52.04	47.59	-	70.76	-
+CPT‡+NTP	46.58	45.76	24.30	56.48	43.28	-4.31	68.18	6.44×10^{19}
+NTP	47.60	50.11	42.60	60.47	50.19	+2.60	70.97	1.72×10^{18}
+NDP	49.49	50.68	42.10	59.95	50.83	+3.24	**71.00**	1.71×10^{18}
+NDP†	**49.49**	**50.83**	**43.70**	**61.22**	**51.25**	**+3.66**	70.99	1.71×10^{18}
LLaMA3	33.70	36.22	2.50	46.98	29.85	-	65.57	-
+CPT‡+NTP	25.29	37.25	10.00	48.76	30.33	+0.48	58.43	6.62×10^{19}
+NTP	31.26	33.09	13.40	39.25	29.25	-0.60	54.09	1.75×10^{18}
+NDP	20.27	27.40	**53.7**	22.99	31.09	+1.24	54.77	1.74×10^{18}
+NDP†	**38.41**	**39.61**	31.6	**50.36**	**40.00**	**+10.15**	**58.58**	1.74×10^{18}

Table 3: Results on domain adaptation task. Item marked with † represents enhancement using PubMed, while ‡ means Pubmed+Redpajama.

(Cettolo et al., 2017) as the training set. Both IWSLT17 and WMT22 (Kocmi et al., 2022) were used as test sets, as IWSLT17 consists of TED talk utterance transcripts while WMT22 comprises news articles. We used WMT22 to observe the generalization performance on out-of-domain data. We use rule-based SacreBLEU (Post, 2018) and neural network based COMET22 (Rei et al., 2022) as evaluation metrics.

Results The result of the translation experiments is shown in Table 2. Overall, NDP consistently outperformed NTP in both in-domain and out-of-domain performance except COMET on T5-small. This suggests that NDP also has considerable potential in small models and downstream-specific tasks.

4.3 Unifying Continue Pre-training and Fine-tuning

Post-training of large language models often includes continued pertaining (CPT), IFT, and RLHF. Here, we focus on CPT and IFT since they involve NTP. Post-training is usually a delicate and complex process because the goal is not only to adapt to a specific domain or align with humans but also to ensure that the knowledge learned during pre-training is minimally disrupted. The NDP offers an optional approach. We still use the dataset from the IFT phase to generate the prefix table, but we additionally use the dataset from the CPT phase to enrich the CLM prefix table, making the resulting CLM distribution incorporates information from the CPT tasks and become more robust. Our method has the following three potential benefits in unifying CPT and IFT: 1) it avoids the cumbersome hyperparameter selection during the continued pre-training phase, such as learning rate, decay, and warmup. 2) by forgoing the continued pre-training phase, the number of model update steps is greatly reduced, which helps alleviate the problem of model forgetting. 3) it saves a lot of training resources since the model still computes only the original instruction dataset.

We have selected two common scenarios: *language transfer* and *domain adaptation*.

Language Transfer We extracted 500k sentences from the monolingual German data in WMT23 (Kocmi et al., 2023) to supplement the CLM distribution. We use T5-xl and T5-Large as the models for this experiment. The COMET result is shown in Figure 3, while BLEU result in Appendix F.

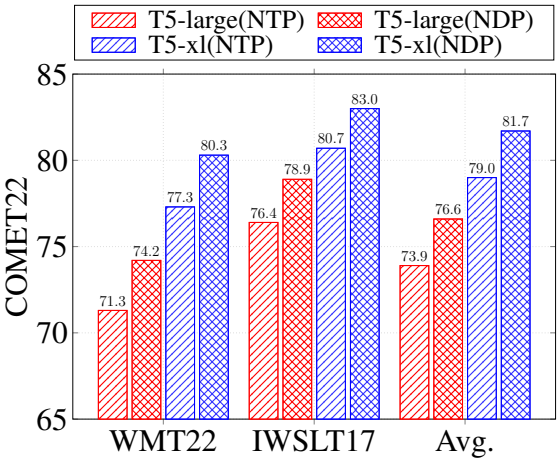

Figure 3: Comparison of COMET22 scores for different models on WMT22 and IWSLT2017 datasets

Our approach achieved a gain of +2.72 on T5-large model and +2.64 on the t5-XL model compared to NTP on average. The BLEU score increased even more, with a gain of +3.18 on the T5-large model and +2.68 on the T5-XL model, confirming the significant benefits of NDP in unifying CPT and IFT. For comparison, similar attempts have been made by NLLB (Team and others), which employed an encoder-decoder model with a Denoising Autoencoder (DAE) as the pretraining task. This task, akin to a cloze test, is simpler than CLM. Additionally, NLLB performs unifying at a step-wise granularity, alternating between fully computing the DAE and the fine-tuning loss in separate steps. In contrast, ours integrates the losses from both the CLM task and the supervised task at a loss-wise level within each iteration, providing a more fine-grained approach.

Domain Adaptation Our methods also show strong performance in vertical domain. We choose PubMed_Abstract which sampled from pile (Gao et al., 2020) and Redpajama-1B as CPT dataset and Alpaca_GPT4+Medquad (Ben Abacha and Demner-Fushman, 2019) as IFT dataset. Test on following benchmarks: MedQA (Jin et al., 2021), MedMCQA (Pal et al., 2022), CareQA (Gururajan et al., 2024), MMLU (Hendrycks et al., 2021a), and

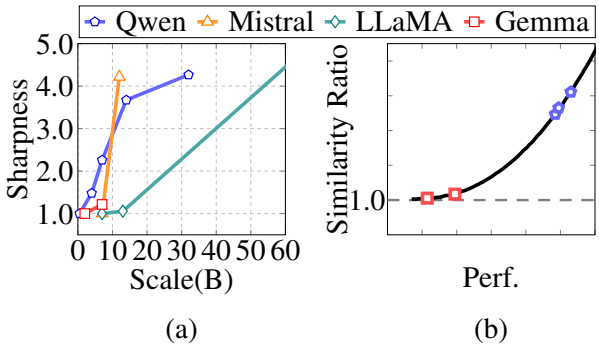

Figure 4: (a): Changes in the sharpness of model distributions with increasing model size. (b): Changes in $\texttt{Sim}_{statistic}/\texttt{Sim}_{ntp}$ with calibrated preformance increases.

PubMedQA (Jin et al., 2019). We retain MMLU to observe its impact on the general domain indirectly. We do full parameter tuning on Qwen2-7B (Bai et al., 2023) and LLaMA3-8B. The result is shown in Table 3, and we can observe that Qwen2 has undergone more adaptation in the medical field compared to LLaMA3. The key findings are as follows:

- For models that lack domain-specific pretraining (such as LLaMA3), NTP leads to a performance drop. In contrast, NDP maintains a steady performance increase.
- The advantages of domain data augmentation are more evident in models without extra domain-specific pre-training. Specifically, Qwen2 exhibits an improvement of +3.66, while LLaMA3 shows a significant increase of +10.15. This suggests that our method holds substantial potential for enhancing the unified continued pretraining process. Moreover, CPT benefits LLaMA3 but negatively impacts Qwen2.
- Across all settings observed in the MMLU benchmark, models trained with NDP not only show superior domain adaptation but also match the general capabilities of models trained with NTP.

More training details can be found in Appendix H.

5 Analysis

Similarity between Training Paradigm and LLM We use LLM distributions as a proxy for the ideal statistical distribution of the world data, since LLM can be seen as an efficient compression

of world data (Deletang et al., 2024). By comparing the similarities between statistic distribution and one-hot distribution with LLM distribution on the same specific datasets, we demonstrate that statistic distribution serves as a superior learning target since it aligns more with LLM distribution.

We observe the cosine similarity between statistical distributions, the one-hot distribution of NTP, and the distribution of LLM, thereby demonstrating that statistical distributions are more efficient as learning targets than NTP distributions.

Since both statistical distributions and LLM distributions have non-one-hot properties, readers might find it unsurprising that statistical distributions are closer to LLM distributions. However, the smooth distribution of LLM is essentially a sharp distribution that is very similar to a one-hot distribution, as shown in Figure 4(a). It is easy to observe that the LLM distribution is not only close to a one-hot distribution, but also that the sharpness[2] of the distribution increases further as the model size grows. To more intuitively show the result, we present the ratio of similarity between the statistical distribution and LLM $\texttt{Sim}_{statistic}$, and the similarity between the NTP distribution and LLM $\texttt{Sim}_{NTP}$, as a function of model performance[3] in Figure 4(b). We observed that as the model's performance improves, the ratio of the statistical distribution to the NTP distribution also increases rapidly. This demonstrates that even as the model size grows, leading to sparser and more concentrated distributions, the statistical distribution still exhibits an essence that is closer to the world data distribution modeled by the LLM.

The Convergence Endpoints of NDP and NTP NDP demonstrates a notable advantage over NTP, however, the source of this superiority, whether from faster convergence or a superior convergence endpoint remains unclear. To investigate long-term convergence behavior, we extended training to *10,000 epochs*.

Drawing from scaling law principles (Kaplan et al., 2020), we use small-scale scenarios to infer large-scale behavior. We trained a *randomly initialized* 438M LLaMA-like model (Ren et al., 2024) on a custom dataset devoid of real-world semantics as shown in Figure 6. This approach eliminates pre-training knowledge effects, allowing pure comparison of NDP and NTP methods. The dataset comprises target items, noise items sharing prefixes with targets, and unrelated items. Noise items simulate real-world interference to target item, while unrelated items help detect overfitting.

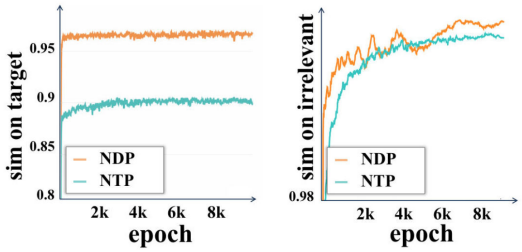

Figure 5: Analysis of model convergence with increasing training epochs. The left figure shows the similarity of the model's output distribution on the target items. The right figure shows the similarity with irrelevant items.

Results are presented using similarity between target frequency distributions as a metric, which correlates with loss. Figure 5 illustrates that NDP's improved fitting accuracy likely stems from a better convergence endpoint, as NTP fails to close the similarity gap after 10,000 epochs. Both methods achieve over 98% fitting accuracy on unrelated items, with NDP showing faster initial convergence. These findings suggest that NDP not only converges more rapidly than NTP but also reaches a superior convergence point.

6 Conclusion

Our work offers a novel critical perspective on the NTP training paradigm, and this hypothesis was validated through preliminary similarity experiments. Based on addressing this issue, we proposed a new training paradigm inspired from statistic called NDP, which achieved good gains in various tasks such as general capability baselines for LLMs, translation, language adaptation, and domain adaptation. Nevertheless, we believe that NDP is merely a simple solution to the narrow candidate problem, and there remains a broad solution space worth exploring to further mitigate this issue.

7 Limitations

Although theoretically, NDP could replace the NTP-based pretraining from scratch and become more powerful as the dataset size increases (because the statistical distribution becomes more ro-

[2]We employ two metrics: one is the proportion of elements of the distribution to reach top-p. The other is kurtosis, which we placed in Appendix A.2. show similar result

[3]We choose the average score list on the Hugging Face Open LLM leaderboard as the performance metric

bust), we lack the resources for practical verification. Therefore, our setup primarily focuses on the instruction fine-tuning process or comparisons with continued pretraining. As such, this remains an area worth exploring.

References

Gregor Bachmann and Vaishnavh Nagarajan. 2024. The pitfalls of next-token prediction. In *ICLR 2024 Workshop: How Far Are We From AGI*.

Jinze Bai et al. 2023. Qwen technical report. *arXiv preprint arXiv:2309.16609*.

Asma Ben Abacha and Dina Demner-Fushman. 2019. A question-entailment approach to question answering. *BMC Bioinformatics*, 20(1).

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *Preprint*, arXiv:2401.10774.

Mauro Cettolo, Marcello Federico, Luisa Bentivogli, Jan Niehues, Sebastian Stüker, Katsuhito Sudoh, Koichiro Yoshino, and Christian Federmann. 2017. Overview of the IWSLT 2017 evaluation campaign. In *Proceedings of the 14th International Conference on Spoken Language Translation*, pages 2–14, Tokyo, Japan. International Workshop on Spoken Language Translation.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *Preprint*, arXiv:2302.01318.

Gregoire Deletang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2024. Language modeling is compression. In *The Twelfth International Conference on Learning Representations*.

Abhimanyu Dubey et al. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Fei Gao, Jinhua Zhu, Lijun Wu, Yingce Xia, Tao Qin, Xueqi Cheng, Wengang Zhou, and Tie-Yan Liu. 2019. Soft contextual data augmentation for neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5539–5544, Florence, Italy. Association for Computational Linguistics.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

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. 2023. A framework for few-shot language model evaluation.

Fabian Gloeckle, Badr Youbi Idrissi, Baptiste Roziere, David Lopez-Paz, and Gabriel Synnaeve. 2024. Better & faster large language models via multi-token prediction. In *Forty-first International Conference on Machine Learning*.

Ashwin Kumar Gururajan, Enrique Lopez-Cuena, Jordi Bayarri-Planas, Adrian Tormos, Daniel Hinjos, Pablo Bernabeu-Perez, Anna Arias-Duart, Pablo Agustin Martin-Torres, Lucia Urcelay-Ganzabal, Marta Gonzalez-Mallo, Sergio Alvarez-Napagao, Eduard Ayguadé-Parra, and Ulises Cortés Dario Garcia-Gasulla. 2024. Aloe: A family of fine-tuned open healthcare llms. *Preprint*, arXiv:2405.01886.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021a. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021b. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. 2015. Distilling the knowledge in a neural network. *Preprint*, arXiv:1503.02531.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. The platonic representation hypothesis. *Preprint*, arXiv:2405.07987.

Di Jin, Eileen Pan, Nassim Oufattole, Wei-Hung Weng, Hanyi Fang, and Peter Szolovits. 2021. What disease does this patient have? a large-scale open domain question answering dataset from medical exams. *Applied Sciences*, 11(14):6421.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Saldyt, and Anil Murthy. 2024a. Llms can't plan, but can help planning in llm-modulo frameworks. *Preprint*, arXiv:2402.01817.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024b. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *Preprint*, arXiv:2001.08361.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Philipp Koehn, Benjamin Marie, Christof Monz, Makoto Morishita, Kenton Murray, Makoto Nagata, Toshiaki Nakazawa, Martin Popel, Maja Popović, and Mariya Shmatova. 2023. Findings of the 2023 conference on machine translation (WMT23): LLMs are here but not quite there yet. In *Proceedings of the Eighth Conference on Machine Translation*, pages 1–42, Singapore. Association for Computational Linguistics.

Tom Kocmi, Rachel Bawden, Ondřej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Thamme Gowda, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Rebecca Knowles, Philipp Koehn, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Michal Novák, Martin Popel, and Maja Popović. 2022. Findings of the 2022 conference on machine translation (WMT22). In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 1–45, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Zuchao Li, Rui Wang, Kehai Chen, Masso Utiyama, Eiichiro Sumita, Zhuosheng Zhang, and Hai Zhao. 2020. Data-dependent gaussian prior objective for language generation. In *International Conference on Learning Representations*.

Xize Liang, Chao Chen, Shuang Qiu, Jie Wang, Yue Wu, Zhihang Fu, Zhihao Shi, Feng Wu, and Jieping Ye. 2024. Ropo: Robust preference optimization for large language models. *Preprint*, arXiv:2404.04102.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2018. Focal loss for dense object detection. *Preprint*, arXiv:1708.02002.

Jiacheng Liu, Sewon Min, Luke Zettlemoyer, Yejin Choi, and Hannaneh Hajishirzi. 2024. Infini-gram: Scaling unbounded n-gram language models to a trillion tokens. *Preprint*, arXiv:2401.17377.

Luca Malagutti, Andrius Buinovskij, Anej Svete, Clara Meister, Afra Amini, and Ryan Cotterell. 2024. The role of n-gram smoothing in the age of neural networks. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6882–6899, Mexico City, Mexico. Association for Computational Linguistics.

Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. *Preprint*, arXiv:2311.13581.

OpenAI, , et al. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa: A large-scale multi-subject multi-choice dataset for medical domain question answering. In *Proceedings of the Conference on Health, Inference, and Learning*, volume 174 of *Proceedings of Machine Learning Research*, pages 248–260. PMLR.

Baolin Peng, Chunyuan Li, Pengcheng He, Michel Galley, and Jianfeng Gao. 2023. Instruction tuning with gpt-4. *arXiv preprint arXiv:2304.03277*.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Ricardo Rei, José G. C. de Souza, Duarte Alves, Chrysoula Zerva, Ana C Farinha, Taisiya Glushkova, Alon Lavie, Luisa Coheur, and André F. T. Martins. 2022. COMET-22: Unbabel-IST 2022 submission for the metrics shared task. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 578–585, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Siyu Ren, Zhiyong Wu, and Kenny Q. Zhu. 2024. EMO: EARTH MOVER DISTANCE OPTIMIZATION FOR AUTO-REGRESSIVE LANGUAGE MODELING. In *The Twelfth International Conference on Learning Representations*.

Adriana Romero, Nicolas Ballas, Samira Ebrahimi Kahou, Antoine Chassang, Carlo Gatta, and Yoshua Bengio. 2015. Fitnets: Hints for thin deep nets. *Preprint*, arXiv:1412.6550.

Dylan Sam and J. Zico Kolter. 2023. Losses over labels: Weakly supervised learning via direct loss construction. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(8):9695–9703.

Gemma Team et al. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

NLLB Team et al. No language left behind: Scaling human-centered machine translation.

Jiaheng Wei, Hangyu Liu, Tongliang Liu, Gang Niu, and Yang Liu. 2022. To smooth or not? when label smoothing meets noisy labels. In *ICML*.

Jingxuan Wei, Linzhuang Sun, Yichong Leng, Xu Tan, Bihui Yu, and Ruifeng Guo. 2024. Sentence-level or token-level? a comprehensive study on knowledge distillation. *Preprint*, arXiv:2404.14827.

Miao Xiong, Zhiyuan Hu, Xinyang Lu, YIFEI LI, Jie Fu, Junxian He, and Bryan Hooi. 2024. Can LLMs express their uncertainty? an empirical evaluation of confidence elicitation in LLMs. In *The Twelfth International Conference on Learning Representations*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions. *Preprint*, arXiv:2304.12244.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2024. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing. *ArXiv*, abs/2406.08464.

Mengyang Yuan, Bo Lang, and Fengnan Quan. 2023. Student-friendly knowledge distillation. *Preprint*, arXiv:2305.10893.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. 2024. Opencodeinterpreter: Integrating code generation with execution and refinement. *Preprint*, arXiv:2402.14658.

A Supplementary of preliminary experiment

A.1 Details

A.1.1 Model & Dataset

In our preliminary experiments, we selected open-source LLMs with varying parameter counts from several series, including: LLaMA, Qwen, Yi, Gemma, and Mistral. For our dataset, We utilized a combination of Alpaca-GPT4, MATH, and CodeFeedback-Filtered-Instruction, as this is a commonly used instruction-tuning dataset combination in engineering (general capabilities + mathematics + code). From this composite dataset, we proportionally sampled 10,000 examples to form our experimental corpus. All data processing codes used in this study are available in our repository for direct access and replication.

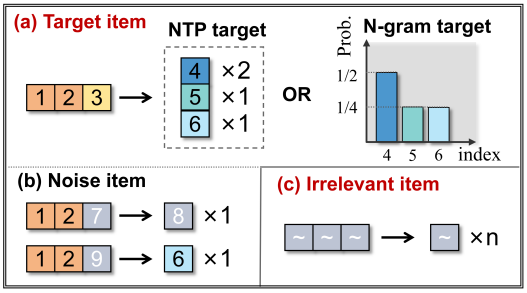

Figure 6: Dataset configuration. Numbers represent tokens, and tilde represents a token that does not repeat with other tokens. The items marked in red font indicate that we will observe its fitting accuracy. ■ are used to represent the common prefix of the input, ■ represent the different suffixes of the input, the blue blocks represent the target to be predicted, and ■ represent the irrelevant tokens. $n = 40$ in our setting.

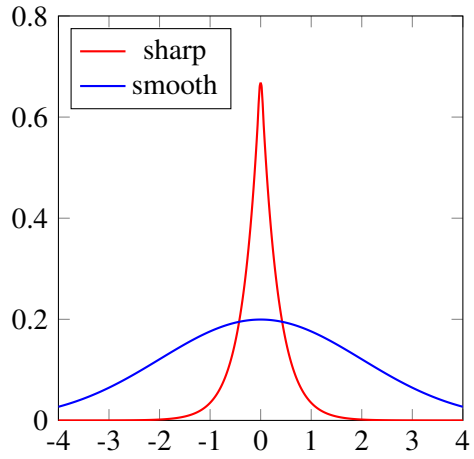

Figure 7: An intuitive example of sharp and smooth distributions.

A.1.2 Calculate similarity

A.2 Dive into sharpness

To illustrate the difference between sharp and smooth distributions more intuitively, we provide an example in Figure 7. This sharpness can be well measured by two metrics, one is the percentage of distribution elements required to achieve a specific probability p as mentioned in the main text, and the other is kurtosis. The previous metric is easy to understand because a sharp distribution has more

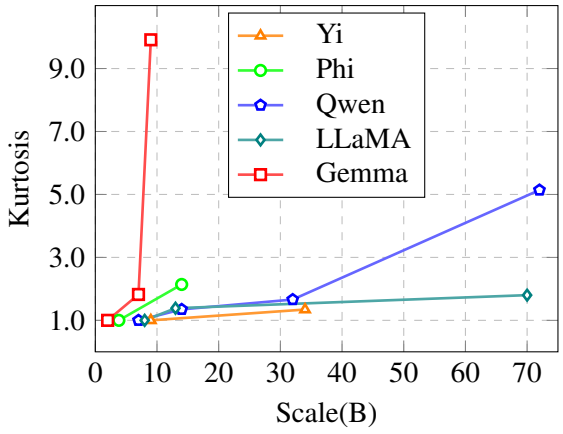

Figure 8: Change in model distribution kurtosis with increasing model size.

probability concentrated in a small number of elements, so there will be fewer elements required to achieve a specific probability. Therefore, the smaller the value, the sparser the distribution. Kurtosis is a commonly used metric in mathematics, used to measure the proportion of probability assigned to the "peaks" in a distribution. We present the changes in kurtosis of different large models as the number of parameters increases in Figure 8. It is easy to observe that for various series of models, kurtosis also increases with the increase of model size, indicating a sharpening of the output distribution of the model.

Although it's somewhat off-topic, what does increased sharpness mean for a model? This actually indicates an overconfidence in the model. Overconfidence can cause the model to hallucinate, producing answers even for questions it's uncertain about. It also exacerbates the phenomenon of error propagation. If the model displays high confidence even for uncertain information, it may lead to further spread of misinformation. It reduces the model's interpretability and credibility. Overly confident models struggle to provide reasonable uncertainty estimates, which decreases the interpretability of their outputs and users' trust in the model. It affects the reliability of decision-making. If the model shows high confidence even in incorrect answers, relying on these confidence estimates for decision-making becomes unreliable (Xiong et al., 2024).

The increase in large language model sharpness is likely attributable to the enhanced memorization capacity that accompanies the growth in model parameters. Consequently, these models can effortlessly retain one-hot features extracted from the dataset by the training paradigm.

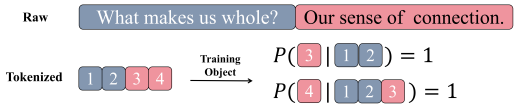

Figure 9: We tokenize the RAW text and observe its true learning objective under the NTP paradigm.However, during implementation, different learning objectives can obtain their corresponding losses through a single forward computation by attention mechanism.

B Distribution from Training Paradigm

The notion of deriving distributions from datasets using training paradigms may be somewhat confusing to readers. In reality, training paradigms can transform each prefix of every instance in a dataset into a distribution. This process can be represented as Figure 9. We can further formalize this process as Equation 4.

$$p(x_{i(j+1)}|x_{i[0:j]}) = 1 \tag{2}$$

Where x_i means the i-th instance in dataset, $x_{i[0:j]}$ means $\{x_{i0}, x_{i1}..x_{ij}\}$. The model's training process on the dataset is equivalent to knowledge distillation from the distribution derived by the NTP training paradigm to the model. In this case, the loss function is simply the Kullback-Leibler (KL) divergence between the NTP distribution and the LLM (Large Language Model) distribution. From this perspective, the NTP training paradigm appears capable of becoming a language model (LM) through the use of a dataset. However, they are not entirely equivalent. This distinction arises when prefixes in the dataset overlap. For instance, when $x_{kl} = x_{ml}$, it becomes unclear whether the distribution should be derived from $x_{k(l+1)}$ or $x_{m(l+1)}$. Understanding the fundamental nature of how NTP derives distributions from datasets allows us to distinguish clearly between NTP-derived distributions and those generated by LLMs.

C Computing and Storage Resources between NTP and NDP

Our implementation is a hybrid of both approaches to achieve a good engineering trade-off. Before training, we precompute a prefix tree for the training set and save a successor word counter of size N, where N is the number of tokens in the training set. During training, we dynamically convert the counter into a discrete distribution.

- **Regarding storage:** When constructing the prefix tree, we observed that the number of

overlapping prefix tokens is relatively small, even less than 1%. We store successor token counters only for the overlapping parts, while other parts store a single successor token as in NTP. In Python, a `Counter` object with 10 items occupies approximately 300 bytes. Thus, the storage size can be estimated as:

$$4N + 0.01N \times 300 \approx 7N \tag{3}$$

For NTP, storing the original text requires saving it in character form. As noted in this work, the Llama3 tokenizer achieves a compression ratio of about 4.61 for English, meaning the character count is approximately $4.61N$, leading to a storage size of $4.61N$ bytes.

- **Regarding training time overhead:** Our model's forward pass speed/FLOPS is identical to that of the NTP model. The only additional step is converting each token's counter into a distribution. This step is computationally efficient and can be effectively masked by leveraging a multi-threaded dataloader (`num_workers=16`) and dataset prefetching (`prefetch_factor=3`).

- **Regarding CPU memory** NDP utilized a Trie structure for constructing the prefix table, storing a Token (int32) for non-overlapping parts and a Counter for overlapping parts. The overlapping sections accounted for roughly 1%, resulting in a memory overhead of approximately $4N$ bytes (original text) + $0.01N \times 300$ bytes (overlapping counters) $\approx$ $7N$ bytes. Each Trie node contains a dict for efficient traversal, which adds an estimated 200 bytes. However, overlapping prefixes share a common path in the Trie, leading to some memory savings. The theoretical upper bound for memory overhead is approximately $207N$ bytes.

- **Potential Directions for Memory Improvement**. We also find suffix-array based methods, such as those in (Liu et al., 2024), which claim to process 1.4T tokens with 10TB of memory. Our method can adapt to such statistical techniques, which could be explored further in the future.

D Convert Frequency to Probability on vocabulary tensor

It is important to note that the vocabulary's dimension is typically much larger than the number of items in the Counter, for instance, 256k vs. 10. If we directly form a probability distribution on such a frequency vector, it would result in a uniform distribution, diluting the information derived from the dataset. We solve this problem by controlling the probability allocated to the zero regions of the tensor.

Without loss of generality, we rearrange a vocab tensor $v = [a_1, a_2 ... a_{|V|}]$ with $|V|$ elements into two contiguous regions based on whether the elements are zero or non-zero. This rearrangement results in $v' = [a'_1 ... a'_k, a'_{k+1} ... a'_{|V|}]$, where $[a'_1 ... a'_k]$ represents the non-zero elements region, and $[a'_{k+1} ... a'_{|V|}]$ represents the zero elements region. Therefore, the softmax process on v' can be described as Equation 4.

$$\texttt{Softmax}(v') = \frac{\sum_{i=1}^{k} e^{a'_i}}{\sum_{j=1}^{|V|} e^{a'_j}} + \frac{\sum_{i=k+1}^{|V|} e^{a'_i}}{\sum_{j=1}^{|V|} e^{a'_j}} \tag{4}$$

where the second item in Equation 4 is the probability value allocated to the entire zero elements region. To control its value and make it equal to our preset probability p, we introduce a temperature coefficient t, transforming it to solve Equation 5.

$$\texttt{f}(t) = \frac{\sum_{i=k+1}^{|V|} e^{a'_i/t}}{\sum_{j=1}^{|V|} e^{a'_j/t}} = p \tag{5}$$

Obtaining an exact solution for Equation 2 is quite challenging; however, we can easily obtain its approximate solution through numerical computation methods. For instance, the root-finding methods provided in scipy[4], or the simpler bisection method, can efficiently locate t within the [0, 100] interval, with the error easily controlled within 1e-6s. Our method is quite different from those commonly used in n-gram language models. The latter can also lead to frequency vectors becoming uniform distributions. For example, the +1 smoothing method could distribute probability values that far exceed the original frequency vector's quantities in a vocabulary of 128k tokens.

[4] https://docs.scipy.org/doc/scipy/tutorial/optimize.html#root-finding

E Settings for General Tasks

E.1 Benchmark Info:

A more detailed description about task type in Table 4:

MC1 (Single-true): Given a question and 4-5 answer choices, select the only correct answer. The model's selection is the answer choice to which it assigns the highest log-probability of completion following the question, independent of the other answer choices. The score is the simple accuracy across all questions.

MC2 (Multi-true): Given a question and multiple true / false reference answers, the score is the normalized total probability assigned to the set of true answers.

Generate: Given a question where the answer is a text snippet, such as code, formulas, or a multiple-choice question that uses Chain of Thought (CoT) for assisted reasoning.

E.2 Train Dataset

As shown in Table 5,We mix them together and shuffle them randomly to create a complete training set. The main reason for choosing these three datasets is that they represent broadly applicable general instructions and highly specialized instructions for mathematics and coding, respectively. This division also reflects the categorization typically found in pre-training processes.

E.3 Training hyperparemeter

Details are listed in Table 6.

F BLEU Score of Language Transfer Task

Result is shown in Figure 10. Although our improvement on this metric is more significant, we have placed it in the appendix, primarily considering that machine translation researchers currently favor using neural network-based evaluation methods to assess translation quality, as rule-based methods might underestimate the performance of large models.

G Drop of LLaMA3 Models.

NTP and NDP Both show a slight decline compared to the LLaMA3-base model. We believe the possible reasons for this are: the pre-training data of LLaMA3-8b amounts to an astonishing 15T tokens, and despite some deduplication, there is still a significant possibility of data leakage in the benchmark. The same phenomenon also appeared in the work of (Xu et al., 2024), which listed results where fine-tuning with various mainstream instruction data caused a decline in benchmark performance.

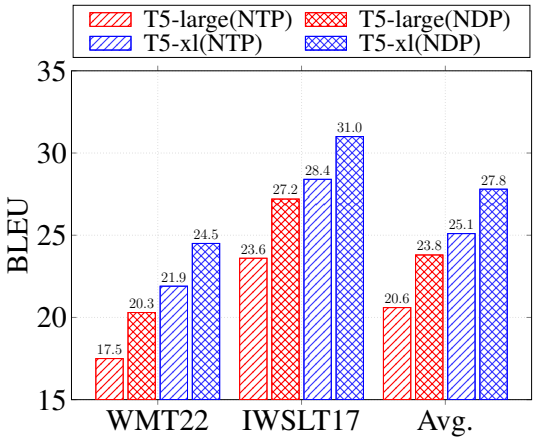

Figure 10: Comparison of BLEU scores for different models on WMT22 and IWSLT2017 datasets

H Training Details in Domain Adaptation Task

Basically, we used shared hyperparameter settings for NTP and NDP as in Table 7.

Dataset name	HF identifier	Type	COT	n	Samples
MMLU	cais/mmlu	MC1		5	14000
GSM8K	openai/gsm8k	Generate	✓	8	1320
HumanEval	openai/openai_humaneval	Generate			164
TruthfulQA	truthfulqa/truthful_qa	MC1(MC12)			817
BBH	lukaemon/bbh	Generate	✓	3	6510
ARC-Challenge	allenai/ai2_arc	MC1			1170
TriviaQA	mandarjoshi/trivia_qa	MC1			17200
AGIEval	RUCAIBox/agieval	MC1			8238
SCIQ	allenai/sciq	MC1			1000
Winogrande	allenai/winogrande	MC1			1767
IFEval	HuggingFaceH4/ifeval	Generate			1080

Table 4: Benchmark used in general task evaluation. n represents instances used for few-shot prompting.

Dataset name	HF identifier	Samples
Alpaca-GPT4	vicgalle/alpaca-gpt4	52000
Math	lighteval/MATH	12500
Code	m-a-p/CodeFeedback-Filtered-Instruction	157000

Table 5: Train dataset used in general task.

Experiment	Hyperparameter name	Setting
NTP	scheduler type	linear
	learning rate	5e-5
	epoch	3
	batch size	512
LS	label smoothing	0.1
NDP	mix ratio	0.8
	n-gram	5

Table 6: Hyperparameter in general task. LS and NDP share the same hyperparameter used in NTP.

Hyperparameter name	Setting
scheduler type	cosine
learning rate	2e-5
CPT epoch	1
IFT epoch	2
batch size	256
cutoff len	8192
warmup ratio	0.05
mix ratio	0.8
n-gram	1

Table 7: Hyperparameter in domain adaption task.