Training LLMs Beyond Next Token Prediction - Filling the Mutual Information Gap

Chun-Hao Yang[1*], Bo-Han Feng[1*], Tzu-Yuan Lai[1], Yan-Yu Chen[1], Yin-Kai Dean Huang[1], Shou-De Lin[1,2]
National Taiwan University[1]
National Taiwan University AI Center of Research Excellence[2]
b10902139@csie.ntu.edu.tw, b10902031@csie.ntu.edu.tw, frogcute122609@gmail.com
b10902137@csie.ntu.edu.tw, dean.huang.tw@gmail.com, sdlin@csie.ntu.edu.tw

Abstract

Autoregressive language models are usually trained with next-token prediction under a fixed left-to-right target order. This paper studies target-token organization as a training-time design choice across arithmetic reasoning, multi-label text classification, and text generation. Controlled order studies show that the default NTP order can be far from the best-performing tested organization, while fixed hand-crafted orders such as reverse order do not generalize across tasks. To address this, **Max(MI($S;t$))** is introduced as a deterministic mutual-information-based heuristic that prioritizes information-rich target tokens. For fixed-length outputs, it constructs task-specific prediction orders; for open-vocabulary text generation, the same principle is applied through lightweight prefix augmentation. Experiments with GPT-style models and pretrained Llama/Qwen variants show that Max(MI($S;t$)) often improves over standard NTP and simple baselines such as reverse order, TF-IDF, and conditional-probability selection, without changing model architecture or autoregressive decoding.

1 Introduction

Large language models (LLMs) can be improved at both test time and training time. Prior work has shown that LLMs are sensitive to the ordering of information presented at test time. Such order effects have been observed across few-shot learning (Kumar and Talukdar, 2021; Lu et al., 2022), reasoning premises (Chen et al., 2024), answer options in multiple-choice evaluation (Pezeshkpour and Hruschka, 2024; Gupta et al., 2024), and retrieved or long-context evidence in RAG-style settings (Liu et al., 2024; Cao et al., 2025). These findings suggest that sequence organization is not merely a formatting detail, but a factor that shapes how autoregressive models use information. However, existing studies primarily manipulate the input context at test time. The analogous question at training time remains underexplored: under a fixed model architecture and standard autoregressive decoding, can learning be improved by organizing the prediction target more effectively before training begins?

Next-token prediction (NTP) is the dominant training objective for LLMs, but it also fixes target prediction to a left-to-right order. This default order may be suboptimal for tasks with latent structure or strong inter-token dependencies.

In this paper, we consider three types of tasks: multi-step arithmetic computations (Shen et al., 2023), multi-label classification (MLC) of text (Zhang et al., 2021; Jaggi et al., 2024; Zhang et al., 2024), and long-form text generation (TG) (Schmidt, 2019; He et al., 2021). In these settings, errors on earlier target tokens may propagate and accumulate. In arithmetic, self-attention must implicitly learn operations such as carry propagation without explicit structural constraints. In MLC, early incorrect labels can bias later predictions toward frequent co-occurrences rather than true label relationships. In TG, early token choices can degrade subsequent coherence, and exposure bias may further amplify this effect. These observations suggest that the default left-to-right order is not always the best target organization for learning.

Take the MLC task in Figure 1 as an example. If an LLM is trained to output information-rich labels first (*insult* label before *obscene* label), it tends to have lower uncertainty when predicting those tokens. Correctly predicting such high-information tokens can, in turn, provide stronger guidance for subsequent predictions. In contrast, NTP, a strict left-to-right generation objective, is more difficult for LLMs to learn, as it does not explicitly prioritize informative tokens and may propagate uncertainty

[*] Equal contribution.

Proceedings of the 15th International Joint Conference on Natural Language Processing and the 5th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics, pages 5539–5554
November 6-10, 2026 ©2026 Association for Computational Linguistics

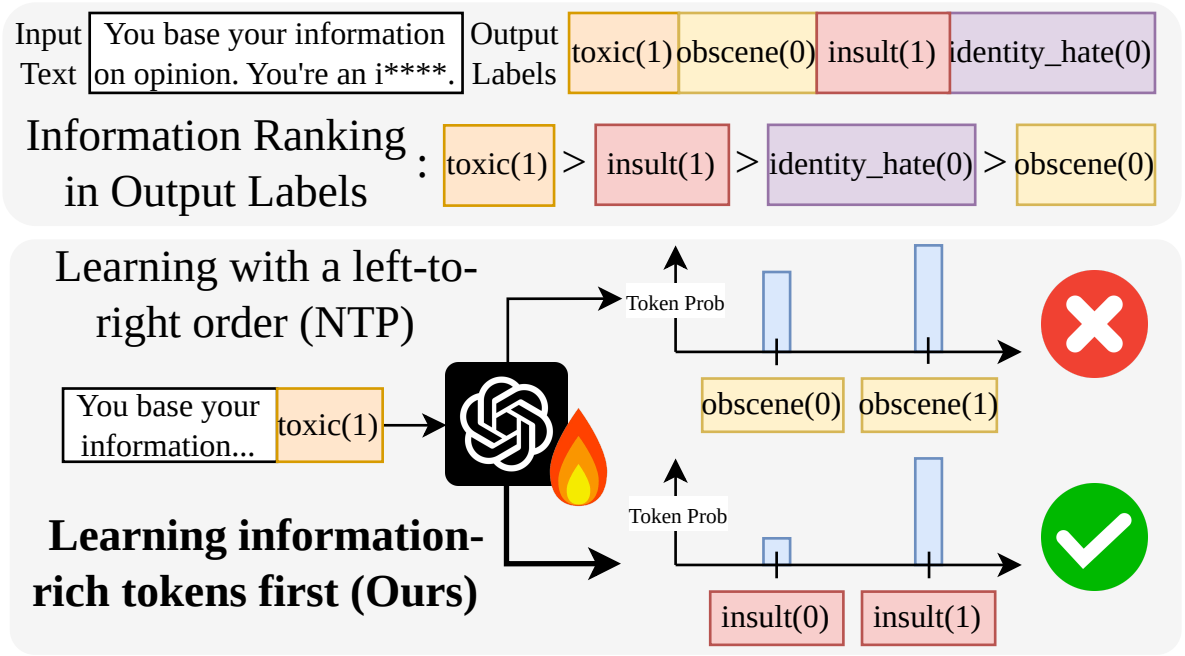


Figure 1: An example of learning information-rich tokens first in MLC task. **Prioritizing information-rich tokens** (*insult* label) during training reduces prediction uncertainty and improves subsequent label prediction, whereas **left-to-right NTP** can be harder to learn due to a lack of emphasis on informative tokens and cumulative uncertainty.

step by step.

The work most closely related to ours addresses this limitation of NTP in arithmetic and shows that reversing the output order can help small transformers on addition tasks (Lee et al., 2024; Nogueira et al., 2021). While insightful, reverse order is still a hand-crafted, task-specific strategy motivated by the standard algorithm of addition, and it does not provide a unified principle for choosing prediction orders across different arithmetic tasks or more general NLP tasks.

To address the suboptimality of NTP and the limited generality of fixed reverse ordering, we propose **Max(MI($S;t$))**, a deterministic strategy prioritizing information-rich target tokens according to their Mutual Information (MI) with the source and previously selected targets. For arithmetic and MLC, where the number of output tokens is defined in advance, **Max(MI($S;t$))** iteratively selects the most informative remaining target token to construct a task-specific prediction order. For TG, it selects one information-rich word for each sentence and duplicates it as a prefix through preprocessing-time augmentation. Both implementations follow the same principle: expose information-rich target content earlier during training. Importantly, our approach modifies only the training target organization with model architecture and standard autoregressive inference unchanged.

Figure 2 shows that no single fixed order generalizes across four representative settings. For arithmetic and MLC in Figure 2 (a)–(c), where each output contains four target digits or labels, we exhaustively evaluate all $4! = 24$ prediction orders. The default NTP order is often far from optimal, ranking 18th, 20th, and 15th in 3-digit addition, 2-digit multiplication, and MLC, respectively. Reverse order matches the best order for addition, but its advantage quickly diminishes in multiplication and MLC, where it ranks 4th and 13th. These results suggest that the reverse order is task-dependent, not a general solution. In contrast, proposed information-based ordering **Max(MI($S;t$))** remains consistently competitive with 5th, 1st, and 7th ranks in the same three settings, achieving accuracies close to the corresponding optima.

For TG in Figure 2 (d), where exhaustive target permutation and reverse ordering are not applicable, we apply **Max(MI($S;t$))** by a lightweight augmentation that duplicates one selected information-rich word earlier in each sentence. This strategy achieves the lowest perplexity among all baselines, reducing perplexity from 18.98 under standard NTP

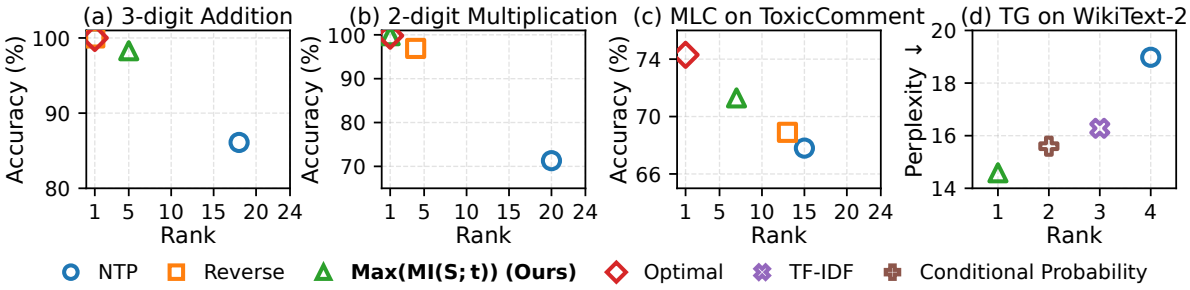


Figure 2: Rank-based comparison of target-token organization strategies. For (a)–(c), we exhaustively evaluate all $4! = 24$ orders for four output digits or labels; NTP ranks poorly among these orders, while Max(MI($S;t$)) consistently approaches or matches the optimum, and reverse order only works well for addition. For (d), where reverse ordering does not apply to natural language generation, we instead expose one selected word earlier by prefix duplication; Max(MI($S;t$)) achieves the lowest perplexity, outperforming NTP, TF-IDF, and conditional-probability baselines. For experimental setup, see Appendix B. For separate illustrative training-curve experiments showing performance discrepancy across training orders, see Figure 9.

to 14.59, and outperforming TF-IDF and conditional-probability-based baselines with the same word selection and prefix duplication procedure. Together, these results indicate that target organization is a substantial training-time factor, and that prioritizing information-rich content offers a more adaptive alternative to fixed left-to-right or hand-crafted reverse ordering.

Contributions. **1. Problem formulation.** We frame target-token organization as a training-time design choice for autoregressive language modeling, showing that left-to-right NTP can be far from the best tested order and that fixed reverse order does not reliably generalize beyond addition. **2. Method.** We propose **Max(MI($S;t$))**, a deterministic MI-based heuristic that prioritizes target tokens most informative with respect to the source and previously selected targets, yielding task-specific prediction orders without changing the model architecture, training objective, or decoding procedure. **3. Open-vocabulary extension.** We extend the same principle to text generation and summarization via a lightweight preprocessing augmentation that exposes selected information-rich words earlier in each sentence. **4. Evaluation and practicality.** Across arithmetic, multi-label classification, WikiText-2 text generation, and XSUM summarization with GPT-style, Llama, and Qwen models, **Max(MI($S;t$))** achieves stronger aggregate performance than NTP, reverse order, TF-IDF, and conditional-probability baselines, with only preprocessing-time changes.

2 Related Work

Prior work has shown that LLMs are sensitive to the ordering of information at inference time. In few-shot learning, changing the order of demonstrations can lead to large performance differences (Kumar and Talukdar, 2021; Lu et al., 2022). Similar effects have been observed for answer options in multiple-choice question answering (Pezeshkpour and Hruschka, 2024; Gupta et al., 2024), premises in reasoning tasks (Chen et al., 2024), and digit order in arithmetic prompting (Singh and Strouse, 2024). Order and position effects also appear in long-context and RAG settings, where models may fail to use evidence depending on where it appears or how retrieved documents are ordered (Liu et al., 2024; Cao et al., 2025). These studies show that sequence organization affects how autoregressive models use information, but they mainly manipulate the input context at test time.

The work most closely related to our studies targets reordering during arithmetic training (Lee et al., 2024; Nogueira et al., 2021). It shows that reversing the output order can improve small transformers on addition, suggesting that the standard left-to-right target order imposed by next-token prediction may be suboptimal. However, reverse order is a hand-crafted strategy motivated by the standard addition algorithm, and its generality beyond addition remains unclear. In contrast, our work studies target organization as a broader training-time design choice across arithmetic, MLC, and TG.

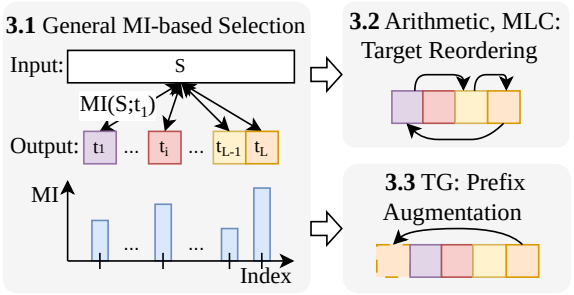


Figure 3: Overview of **Max(MI($S;t$))**. We first compute MI scores between the current source and candidate targets. Fixed-length tasks use the selected targets to form a reordered output sequence, while open-vocabulary TG uses the selected word for per-sentence prefix augmentation.

3 Information-Rich Token Prediction

Not all target tokens are equally informative during training. Some tokens are more strongly determined by the source, and once predicted, can provide useful context for remaining targets. Instead of always following a fixed left-to-right order, our goal is to predict target tokens sharing the most information with the current source earlier. We formalize this idea with MI.

Figure 3 summarizes the method flow. We first score candidate targets by their MI with the current source. For fixed-length outputs such as arithmetic digits and MLC labels, the highest-MI target is appended to the source and the process iterates to produce a reordered target sequence. For open-vocabulary TG, the same criterion selects one information-rich word per sentence for prefix duplication. The following subsections define the general MI-based selection rule and its two instantiations.

3.1 General MI-based Selection

We introduce **Max(MI($S;t$))**, a deterministic strategy that constructs a prediction order by iteratively selecting the remaining target position with the highest MI with the current source. Let $S^{(0)} = [s_1, \cdots, s_N] \in \mathbb{R}^{L_1 \times N}$ denote the initial source sequences, where N is the number of training samples, L_1 is the source length, and each column s_i is the source sequence of the i^{th} sample. Let $T = [t_1, \cdots, t_{L_2}] \in \mathbb{R}^{N \times L_2}$ denote the target tokens, where L_2 is the target length, t_j collects the j^{th} target token across all samples, and t_{ji} is the j^{th} target token of sample i.

At selection step k, let $S^{(k)}$ be the current source after appending the k previously selected target positions to the original source, and let $s_i^{(k)}$ be its value for sample i. For each remaining target position j, let V_j be the set of possible values of t_{ji}. We compute:

$$\mathrm{MI}(S^{(k)}; t_j) = \sum_{s_i^{(k)} \in S^{(k)}} \sum_{t_{ji} \in V_j} P(s_i^{(k)}, t_{ji}) \log \frac{P(s_i^{(k)}, t_{ji})}{P(s_i^{(k)}) P(t_{ji})}. \tag{1}$$

The next target position is selected as:

$$j^* = \arg\max_{j \in R^{(k)}} \mathrm{MI}(S^{(k)}; t_j), \tag{2}$$

where $R^{(k)}$ is the set of remaining target positions. The selected target t_{j^*} is appended to the source to form $S^{(k+1)}$ and removed from $R^{(k)}$. This process repeats until all target positions are ordered. Further theoretical justification is provided in Appendix A.1, and a step-by-step calculation example is given in Appendix A.2.

3.2 Fixed-Length Target Reordering

Arithmetic and MLC have fixed-length targets, making explicit target reordering feasible. Arithmetic tasks predict output digits, while MLC tasks predict a sequence of label decisions. For a 2-digit multiplication task ($A_1A_2 \times B_1B_2 = C_1C_2C_3C_4$), the source is $A_1A_2B_1B_2$ and the candidate target positions are C_1, C_2, C_3, C_4. If

$$\arg\max_j \mathrm{MI}(\mathit{source}; C_j) = 3, \tag{3}$$

then C_3 is predicted first and added to the source for the next selection step. The same procedure applies to MLC by replacing digit positions with label positions: the label with the highest MI with the input text is predicted first, then used as source for selecting subsequent labels. Algorithm 1 in Appendix A.3 summarizes this procedure formally.

For arithmetic and MLC, MI is computed from empirical counts over a small discrete target space. Thus, preprocessing only requires an offline counting-and-sorting pass on a CPU and does not require training an auxiliary model.

3.3 Open-Vocabulary Prefix Augmentation

For text generation, target sentences have variable length and open-vocabulary tokens, making exhaustive target reordering impractical. We therefore apply the same information-rich-token

principle through preprocessing-time augmentation: for each target sentence, we select one information-rich word and duplicate it at the beginning of the sentence.

Let $S = (s_1, s_2, \cdots, s_{N_s})$ denote the source context before the target sentence, where N_s is the source-context length, and let $T = (t_1, t_2, \cdots, t_M)$ denote the words in the target sentence, where M is the target-sentence length. Unlike arithmetic and MLC, where $P(s_i, t_j)$ can be estimated directly from finite target spaces, TG involves open-vocabulary text. To approximate MI, we assume a first-order Markov structure where the probability of a target word depends primarily on the previous context token. We train a two-layer logistic regression classifier over token bigrams to estimate the joint probability of observing a target token t_j after a context token s_i.

Let $\mathbf{e}_{s_i} \in \mathbb{R}^d$ and $\mathbf{e}_{t_j} \in \mathbb{R}^d$ be the embedding vectors for tokens s_i and t_j, respectively. We parameterize the estimated joint probability as:

$$P(s_i, t_j) = \sigma\left(\mathbf{w}_2\left(\mathbf{w}_1[\mathbf{e}_{s_i}; \mathbf{e}_{t_j}] + b_1\right) + b_2\right), \tag{4}$$

where $\mathbf{w}_1 \in \mathbb{R}^{h \times 2d}$, $\mathbf{w}_2 \in \mathbb{R}^{1 \times h}$, $b_1 \in \mathbb{R}^h$, and $b_2 \in \mathbb{R}$ are trainable parameters with hidden size h, and σ is the sigmoid function. The model is trained using adjacent token pairs from the training corpus.

For each candidate word t_j in the target sentence T, we define:

$$\text{MI}(S; t_j) = P(S, t_j) \log \frac{P(S, t_j)}{P(S)P(t_j)}. \tag{5}$$

Under the Markov approximation, $P(S, t_j)$ is estimated using the final source-context token s_{N_s}, yielding the ranking score:

$$\text{MI-Score}(t_j) = P(s_{N_s}, t_j) \log \frac{P(s_{N_s}, t_j)}{P(t_j)}, \tag{6}$$

where $P(t_j)$ is the marginal probability of token t_j, obtained by aggregating its joint probabilities with possible context tokens in the corpus. We then select:

$$t^* = \arg\max_{t_j \in T} \text{MI-Score}(t_j). \tag{7}$$

For practical preprocessing cost in TG, MI scoring requires a lightweight auxiliary model because the target space is open-vocabulary. This makes preprocessing more expensive than in arithmetic and MLC, but it remains a one-time offline step; see Appendix B.3 for complexity analysis and empirical runtime.

4 Experiments for Arithmetic Tasks

Arithmetic tasks provide controlled settings for studying target-token organization: output digits have fixed positions, all target-order permutations are enumerable for small tasks, and exact correctness can be measured. We evaluate **Max(MI(S;t))** against **NTP**, the standard left-to-right target order, and **Reverse**, the reversed NTP order. For GPT-based models, we exhaustively test all target-token permutations to identify the best tested order. We further evaluate the same strategies by fine-tuning a billion-parameter Qwen-Math model in both multi-task and long-sequence multiplication settings. Experimental details are provided in Appendix B.1.

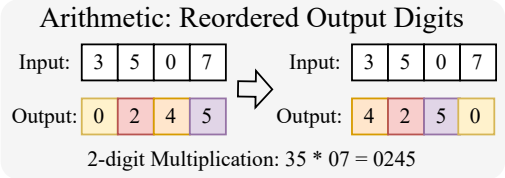


Figure 4: Arithmetic data format with reordered digits. The source expression is fixed, while output digits are rearranged according to the selected prediction order.

4.1 Datasets & Evaluation

We consider addition, multiplication, logarithm, greatest common divisor (GCD), and the chicken-rabbit (CR) problem. As shown in Figure 4, each example consists of a fixed source expression and reordered output digits. For GPT-based experiments, each task is programmatically generated and trained separately; the number of target-order permutations ranges from 3! for 3-digit outputs to 6! for 6-digit outputs.

For example, $\log_{10}(1234) = 3.091$ is represented as $[1, 2, 3, 4, 3, 0, 9, 1]$, with target digits $C_1C_2C_3C_4 = [3, 0, 9, 1]$. For Qwen-Math fine-tuning, we mix all arithmetic tasks used in the GPT-based experiments except 3-digit multiplication, and keep mathematical symbols to match instruction-style arithmetic data, e.g., $379 + 841 = 1270$ is represented as $[3, 7, 9, +, 8, 4, 1, =, 1, 2, 7, 0]$.

We also evaluate long-sequence multiplication with Qwen-Math by multiplying two d-digit numbers for $d = 3, \ldots, 8$, where the output length increases up to 16 digits. Due to the rapidly growing data space, we sample 100,000 examples

Task	Best Order	Best Acc.	NTP	Reverse	Max(MI(S;t))
3-digit Addition	$C_4C_3C_2C_1$	99.99%	85.09%	**99.99%**	97.46%
2-digit Multiplication	$C_3C_2C_4C_1$	98.22%	70.15%	95.35%	**97.38%**
4-digit Logarithm	$C_1C_4C_3C_2$	98.21%	95.99%	**96.63%**†	**96.63%**†
3-digit Greatest Common Divisor	$C_1C_3C_2$	95.80%	91.45%	**92.59%**†	**92.59%**†
3-digit Multiplication	$C_5C_4C_3C_6C_2C_1$	99.66%	71.43%	88.15%	**96.23%**
2-digit Chicken-Rabbit Problem	$C_4C_2C_3C_1$	92.12%	34.92%	45.04%	**89.48%**
Average		97.33%	74.84%	86.29%	**94.96%**

Table 1: GPT-based model performance across different arithmetic tasks. Token prediction strategies: (1) **NTP**: standard left-to-right target order. (2) **Reverse**: reversed NTP. (3) **Max(MI(S;t))**: prioritizes output digits with the highest mutual information with the source. †: Identical predicted token sequences for both strategies.

for each long-sequence setting.

Predicted target orders are converted back to numerical answers. We report fixed-iteration accuracy, where the iteration count is determined by the early plateau of the best-performing order.

4.2 Results

Table 1 shows that target-token order substantially affects arithmetic learning. **Max(MI(S;t))** achieves the highest average accuracy among the three practical strategies, reaching 94.96% compared with 74.84% for NTP and 86.29% for Reverse. It also approaches the exhaustive-search upper bound of 97.33%, indicating that neither left-to-right nor reverse order is universally reliable across arithmetic tasks.

Task	NTP	Reverse	Max(MI(S;t))
3-digit Addition	99.3%	99.8%	**100.0%**
2-digit Multiplication	94.1%	95.4%	**96.7%**
4-digit Logarithm	89.1%	89.0%†	**89.6%**†
3-digit Greatest Common Divisor	94.0%	93.7%†	**94.3%**†
2-digit Chicken-Rabbit Problem	84.7%	96.9%	**99.4%**
Average	92.2%	95.0%	**96.0%**

Table 2: Qwen-Math performance across arithmetic tasks. **NTP**: standard left-to-right order; **Reverse**: reversed NTP order; **Max(MI(S;t))**: prioritizes information-rich output digits. †: Identical predicted token sequences for Reverse and Max(MI(S;t)).

Table 2 shows the same trend for multi-task Qwen-Math fine-tuning. Max(MI(S;t)) achieves the best average accuracy, exceeding NTP by 3.8% and Reverse by 1.0%. Along with the GPT-based exhaustive studies, this suggests Max(MI(S;t)) as a task-adaptive alternative to fixed hand-crafted orders.

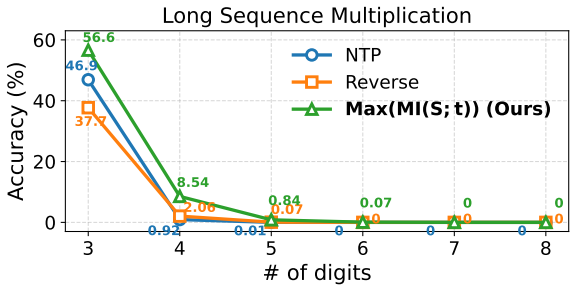


Figure 5: Long-sequence multiplication with Qwen-Math. Each task multiplies two d-digit numbers. All strategies degrade as output length grows, but Max(MI(S;t)) shows better robustness before the longest settings collapse.

Figure 5 further evaluates longer multiplication outputs. All strategies degrade sharply as digit length increases and fail in the longest settings, but Max(MI(S;t)) maintains higher accuracy than NTP and Reverse before this collapse, indicating better robustness under increasing output length.

5 Experiments for Multi-label Classification

MLC provides a structured NLP setting for studying target-token organization: each input text is mapped to a fixed set of label decisions, and label tokens carry semantic information. Unlike multiple-choice question-answering options such as A/B/C/D, MLC labels can express both text-label relevance and dependencies among labels. We evaluate **Max(MI(S;t))** against **NTP** and **Reverse** across multiple datasets, languages, and model families. Experimental details are provided in Appendix B.2.

Model	Dataset	Best Order	Best Acc.	NTP	Reverse	Max(MI(S;t))
Llama-3.1-8B-Instruct	ToxicComment	$C_3C_1C_2C_4$	85.81%	84.22%	82.59%	**85.14%**
Llama-3.2-1B-Instruct	ToxicComment	$C_1C_4C_3C_2$	77.20%	74.65%	74.48%	**75.04%**
Qwen2.5-3B-Instruct	ToxicComment	$C_1C_3C_4C_2$	78.85%	76.24%	70.98%	**78.76%**
Qwen2.5-1.5B-Instruct	ToxicComment	$C_2C_1C_3C_4$	74.26%	67.87%	68.90%	**71.96%**
GPT-2	ToxicComment	$C_3C_1C_2C_4$	88.63%	88.02%	**88.17%**	88.15%
GPT-2	PaperAbstract	$C_4C_1C_2C_3$	88.73%	**88.52%**†	88.23%	**88.52%**†
GPT-2	GoEmotions	$C_2C_3C_4C_1$	96.40%	**96.24%**†	96.18%	**96.24%**†
Average			84.27%	82.25%	81.36%	**83.40%**

Table 3: Maximal accuracy in MLC across models and English datasets. **NTP**: standard left-to-right label order; **Reverse**: reversed NTP order; **Max(MI(S;t))**: prioritizes information-rich label tokens. † indicates identical predicted label sequences.

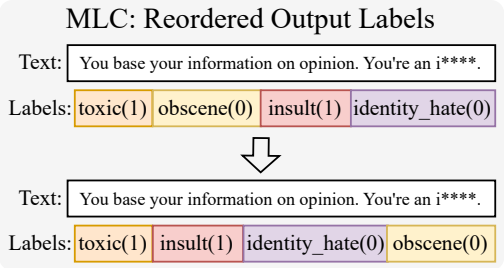


Figure 6: MLC data format with reordered labels. The input text is fixed, while output label decisions are rearranged according to the selected prediction order.

5.1 Datasets & Evaluation

We evaluate on three MLC datasets: *ToxicComment*, *PaperAbstract*, and *GoEmotions*. To control target length, we retain the four most frequent labels in each dataset and denote them as C_1, C_2, C_3, C_4. As shown in Figure 6, each example keeps the input text fixed and reorders only the output label decisions.

For multilingual evaluation, we additionally translate *ToxicComment* into Chinese using Google Translate. This setting tests whether target-order effects persist when the model may have different pretraining exposure to the input language. We report label-level accuracy.

5.2 Results

Table 3 shows that target-label order affects MLC performance across models and English datasets. On average, **Max(MI(S;t))** achieves 83.40% accuracy, outperforming NTP by 1.15% and Reverse by 2.04%. Reverse order is weaker than NTP on average, contrasting with its strong performance in arithmetic and suggesting that fixed hand-crafted orders do not transfer reliably across task types.

Model	NTP	Reverse	Max(MI(S;t))
Llama-3.1-8B†	81.52%	80.16%	**82.82%**
Llama-3.2-1B†	72.84%	72.39%	**74.61%**
Qwen2.5-3B†	72.73%	71.92%	**74.39%**
Qwen2.5-1.5B†	68.08%	**70.39%**	70.18%
GPT-2	75.65%	69.42%	**79.70%**
Average	74.16%	72.86%	**76.34%**

Table 4: Maximal accuracy in MLC on the Chinese *ToxicComment* dataset. **NTP** denotes the standard left-to-right label order; **Reverse** denotes the reversed NTP order; **Max(MI(S;t))** prioritizes information-rich label tokens. † indicates instruction-tuned models.

Table 4 shows the same comparison on Chinese *ToxicComment*. Max(MI(S;t)) again obtains the best average accuracy, improving over NTP by 2.18% and over Reverse by 3.48%. The gains are larger than in the English setting, suggesting that target-token organization may interact with language and pretraining exposure. Overall, MLC results support the need for task-adaptive label ordering rather than a universal fixed order.

6 Experiments for Text Generation

For open-vocabulary text generation, exhaustive target reordering is impractical. We therefore evaluate the prefix-augmentation strategy introduced in Section 3.3: for each sentence, one selected word is duplicated at the beginning of

the sentence. We compare four strategies: **NTP**, the standard unaugmented baseline; **TF-IDF**, which selects the word with the highest TF-IDF score; **P($t|S$)**, which selects the word with the highest conditional probability given the source; and **Max(MI(S;t))**, which selects the most information-rich word. We evaluate these strategies on WikiText-2 continued pretraining using perplexity and on XSUM summarization using ROUGE. Experimental details are provided in Appendix B.3, and additional NLU results are reported in Appendix C.

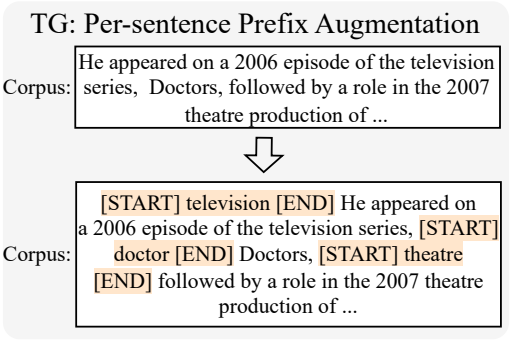


Figure 7: TG data format with per-sentence prefix augmentation. The selected word is inserted before each sentence between `[START]` and `[END]`.

6.1 Datasets & Evaluation

For text generation, GPT-2 and Llama-3.2-1B are continually pretrained on *WikiText-2*. We tokenize and lemmatize the corpus, segment documents into sentences using punctuation, select one word per sentence according to each strategy, map the selected word back to its surface form, and duplicate it at the sentence beginning with `[START]` and `[END]` delimiters, as shown in Figure 7. We evaluate using perplexity. To compare original and augmented data on the same evaluable tokens, we mask the inserted special tokens, selected words, and the immediately following token during perplexity computation.

For summarization, Llama-3.2-1B-Instruct is fine-tuned on *XSUM*. For all augmented strategies, we concatenate document and summary, apply the same per-sentence word selection and prefix augmentation as WikiText-2, and then split the augmented text back into document and summary fields for fine-tuning. We evaluate generated summaries using ROUGE-1, ROUGE-2, and ROUGE-L, with `[START]`, `[END]`, and inserted words removed for fair comparison.

Model	NTP	TF-IDF	P($t|S$)	Max(MI(S;t))
GPT-2	18.98	16.28	15.60	**14.59**
Llama-3.2-1B	8.19	7.51	7.27	**6.95**
Average	13.58	11.90	11.44	**10.77**

Table 5: Perplexity on WikiText-2. Lower is better. NTP denotes the standard unaugmented baseline; TF-IDF, P($t|S$), and Max(MI(S;t)) use the same prefix-augmentation format with different word-selection criteria.

Metric	NTP	TF-IDF	P($t|S$)	Max(MI(S;t))
ROUGE-1	0.2838	0.3298	0.3301	**0.3313**
ROUGE-2	0.0862	0.1093	0.1091	**0.1102**
ROUGE-L	0.2127	0.2547	0.2534	**0.2554**
Average	0.1942	0.2313	0.2309	**0.2323**

Table 6: ROUGE scores on XSUM using Llama-3.2-1B-Instruct. Higher is better.

6.2 Results

Table 5 shows that prefix augmentation reduces WikiText-2 perplexity for both GPT-2 and Llama-3.2-1B. Max(MI(S;t)) performs best, lowering average perplexity from 13.58 under NTP to 10.77, while TF-IDF and P($t|S$) also outperform NTP. This suggests that earlier exposure to selected words benefits continued pretraining, with the MI-based criterion being the strongest strategy.

Table 6 shows a similar trend on XSUM. Max(MI(S;t)) achieves the best ROUGE-1/2/L and average ROUGE scores, though only modestly ahead of TF-IDF and P($t|S$). Overall, information-rich prefix augmentation improves open-vocabulary generation without architectural or decoding changes.

7 Conclusion

We studied target-token organization as a training-time design choice for autoregressive language modeling. Across arithmetic, MLC, and TG, left-to-right NTP can be suboptimal, while fixed hand-crafted orders, such as reverse order, do not consistently generalize. Max(MI(S;t)) prioritizes information-rich target content via fixed-length reordering or open-vocabulary prefix augmentation. Experiments with GPT-style models and pretrained Llama/Qwen variants show that it often improves over NTP and simple baselines, highlighting target organization as a practical direction for LLM training.

Limitations

Our study has several limitations. First, Max(MI(S;t)) is a heuristic for target organization and does not guarantee the optimal prediction order. In fixed-length tasks, the best order can still depend on task-specific structure, such as positional dependencies in arithmetic. The long-sequence multiplication results further show that target organization alone does not solve long-output arithmetic scaling when sequences become very long. In open-vocabulary TG, our method uses prefix augmentation rather than exhaustive reordering, so it only approximates the broader space of possible target organizations.

Second, our experiments focus on decoder-only models from the GPT, Llama, and Qwen families. Although the results cover both models trained from scratch and pretrained variants, the effect of target organization may vary with model scale, architecture, pretraining data, and tokenizer design. Further validation is needed for larger models and other architectures.

Third, the multilingual MLC results suggest that language and pretraining exposure may affect the benefit of target organization. However, our Chinese setting is based on translated ToxicComment data, so it does not fully isolate linguistic effects from translation artifacts or domain differences. Broader multilingual evaluation is needed to better understand these factors.

Finally, Max(MI(S;t)) requires estimating MI between source and target tokens. For arithmetic and MLC, this is inexpensive because the target space is small and discrete; for open-vocabulary TG, it requires an auxiliary offline scoring model. Although the preprocessing cost is a one-time step, more scalable MI approximations would be useful for larger corpora and longer contexts.

Acknowledgments

This material is based upon work supported by National Science and Technology Council, ROC under grant number 114-2221-E-002-134-MY3, NTU AI Center of Research Excellence within Taiwan Centers of Excellence in Artificial Intelligence, and by National Taiwan University and Academia Sinica Innovative Joint Program, under grant AS-NTU-114-06. We acknowledge the use of ChatGPT for assisting in the grammatical review and refinement of this document.

References

Shuyang Cao, Karthik Radhakrishnan, David Rosenberg, Steven Lu, Pengxiang Cheng, Lu Wang, and Shiyue Zhang. 2025. Evaluating the retrieval robustness of large language models. *Preprint*, arXiv:2505.21870.

Xinyun Chen, Ryan Andrew Chi, Xuezhi Wang, and Denny Zhou. 2024. Premise order matters in reasoning with large language models. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 6596–6620. PMLR.

Dorottya Demszky, Dana Movshovitz-Attias, Jeongwoo Ko, Alan Cowen, Gaurav Nemade, and Sujith Ravi. 2020. GoEmotions: A dataset of fine-grained emotions. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4040–4054, Online. Association for Computational Linguistics.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. QLoRA: Efficient finetuning of quantized LLMs. In *Advances in Neural Information Processing Systems*, volume 36, pages 10088–10115. Curran Associates, Inc.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, and others. 2024. The Llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Vipul Gupta, David Pantoja, Candace Ross, Adina Williams, and Megan Ung. 2024. Changing answer order can decrease MMLU accuracy. *Preprint*, arXiv:2406.19470.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James Glass. 2021. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5087–5102, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Harbani Jaggi, Kashyap Coimbatore Murali, Eve Fleisig, and Erdem Biyik. 2024. Accurate and data-efficient toxicity prediction when annotators disagree. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21910–21917, Miami, Florida, USA. Association for Computational Linguistics.

Baoyu Jing, Si Zhang, Yada Zhu, Bin Peng, Kaiyu Guan, Andrew Margenot, and Hanghang Tong. 2022. Retrieval based time series forecasting. *Preprint*, arXiv:2209.13525.

Sawan Kumar and Partha Talukdar. 2021. Reordering examples helps during priming-based few-shot

learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4507–4518, Online. Association for Computational Linguistics.

Nayoung Lee, Kartik Sreenivasan, Jason Lee, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Teaching arithmetic to small transformers. In *International Conference on Learning Representations*, volume 2024, pages 25001–25054.

Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2024. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland. Association for Computational Linguistics.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. *CoRR*, abs/1609.07843.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1797–1807, Brussels, Belgium. Association for Computational Linguistics.

Rodrigo Nogueira, Zhiying Jiang, and Jimmy Lin. 2021. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv preprint arXiv:2102.13019*.

Pouya Pezeshkpour and Estevam Hruschka. 2024. Large language models sensitivity to the order of options in multiple-choice questions. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 2006–2017, Mexico City, Mexico. Association for Computational Linguistics.

Qwen, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, and 24 others. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence Level Training with Recurrent Neural Networks. In *International Conference on Learning Representations*.

Florian Schmidt. 2019. Generalization in generation: A closer look at exposure bias. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 157–167, Hong Kong. Association for Computational Linguistics.

Ruoqi Shen, Sébastien Bubeck, Ronen Eldan, Yin Tat Lee, Yuanzhi Li, and Yi Zhang. 2023. Positional description matters for transformers arithmetic. *Preprint*, arXiv:2311.14737.

Aaditya K. Singh and DJ Strouse. 2024. Tokenization counts: the impact of tokenization on arithmetic in frontier llms. *Preprint*, arXiv:2402.14903.

Liam van der Poel, Ryan Cotterell, and Clara Meister. 2022. Mutual information alleviates hallucinations in abstractive summarization. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5956–5965, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Chaojun Wang and Rico Sennrich. 2020. On exposure bias, hallucination and domain shift in neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3544–3552, Online. Association for Computational Linguistics.

Pinyi Zhang, Jingyang Chen, Junchen Shen, Zijie Zhai, Ping Li, Jie Zhang, and Kai Zhang. 2024. Message passing on semantic-anchor-graphs for fine-grained emotion representation learning and classification. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 2771–2783, Miami, Florida, USA. Association for Computational Linguistics.

Ximing Zhang, Qian-Wen Zhang, Zhao Yan, Ruifang Liu, and Yunbo Cao. 2021. Enhancing label correlation feedback in multi-label text classification via multi-task learning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1190–1200, Online. Association for Computational Linguistics.

A Details of Information-Rich Token Prediction Method

A.1 Theoretical Justification

Assumption. The LLM predictions can be modeled as a Markov chain $I \rightarrow \tilde{T} \rightarrow T$, where

I is the input token, $\tilde{T}$ is the model output token, and T is the ground-truth token. This reflects the autoregressive nature of LLMs, where the next token is sampled conditionally.

Connection to MI. Minimizing mean-square error (MSE) between model predictions $\tilde{T}$ and ground-truth tokens T is equivalent to maximizing the MI$(T;\tilde{T})$:

$$\min_{T} \mathbb{E}[||T-\tilde{T}||^2] \iff \max \text{MI}(T;\tilde{T}) \quad (8)$$

Proof Sketch. This equivalence was demonstrated in Jing et al. (2022), where the prediction objective under a Gaussian assumption leads to maximal MI between the predicted and true values. Intuitively, higher MI reduces uncertainty and improves reconstruction accuracy.

Motivation via DPI. Given the Markov chain $I \rightarrow \tilde{T} \rightarrow T$, and using the Data Processing Inequality (DPI), we obtain:

$$\text{MI}(I;T) \leq \text{MI}(\tilde{T};T) \quad (9)$$

Hence, MI(I;T) can be viewed as a proxy for how much source information is available for predicting the target. This motivates prioritizing target tokens with higher MI with the source.

Implication. During training, prioritizing the target token t with maximum MI with the source input I, i.e., $\arg\max_t \text{MI}(I;t)$, encourages the model to predict more source-informative targets earlier.

A.2 Calculation Example of MI(S;t_j)

We provide a step-by-step demonstration of MI(S;t_j) calculation to clarify our method. Assume there are three 2-digit addition data samples:

- $12+34=046$
- $55+66=121$
- $07+08=015$

S would be:

$$[s_1,s_2,s_3] = \left[\begin{pmatrix}1\\2\\3\\4\end{pmatrix}\begin{pmatrix}5\\5\\6\\6\end{pmatrix}\begin{pmatrix}0\\7\\0\\8\end{pmatrix}\right] \quad (10)$$

In equation 10, s_i is the i^{th} data sample with $L_1=4$ and $N=3$.

T would be:

$$\begin{aligned}[t_1,t_2,t_3] &= \left[\begin{pmatrix}t_{11}\\t_{12}\\t_{13}\end{pmatrix}\begin{pmatrix}t_{21}\\t_{22}\\t_{23}\end{pmatrix}\begin{pmatrix}t_{31}\\t_{32}\\t_{33}\end{pmatrix}\right] \\ &= \left[\begin{pmatrix}0\\1\\0\end{pmatrix}\begin{pmatrix}4\\2\\1\end{pmatrix}\begin{pmatrix}6\\1\\5\end{pmatrix}\right]\end{aligned} \quad (11)$$

In equation 11, t_{ji} is j^{th} target token of i^{th} data sample with $N=3$ and $L_2=3$. V_j for t_j is $\{t_{j1},\ t_{j2},\ t_{j3}\}$.

Therefore, to calculate MI(S; t_1), which is MI($source$, C_1) in Algorithm 1, we can obtain the value of $\text{MI}(S;t_1) = \sum_{s_i\in S}\sum_{t_{1i}\in V_1} P(s_i,t_{1i})\log\frac{P(s_i,t_{1i})}{P(s_i)P(t_{1i})}$ from the summation of the following terms:

- $P(s_i=\begin{pmatrix}1\\2\\3\\4\end{pmatrix}, t_{1i}=0)\log(...)$,
- $P(s_i=\begin{pmatrix}1\\2\\3\\4\end{pmatrix}, t_{1i}=1)\log(...)$,
- $P(s_i=\begin{pmatrix}5\\5\\6\\6\end{pmatrix}, t_{1i}=0)\log(...)$,
- $P(s_i=\begin{pmatrix}5\\5\\6\\6\end{pmatrix}, t_{1i}=1)\log(...)$,
- $P(s_i=\begin{pmatrix}0\\7\\0\\8\end{pmatrix}, t_{1i}=0)\log(...)$,
- $P(s_i=\begin{pmatrix}0\\7\\0\\8\end{pmatrix}, t_{1i}=1)\log(...)$.

A.3 Pseudocode for Information-Rich Token Selection

Algorithm 1 Information-Rich Token Selection

Input: source tokens $[A_1, A_2, B_1, B_2]$, target tokens $[C_1, C_2, C_3, C_4]$
Output: reordered token sequence

$source \leftarrow [A_1, A_2, B_1, B_2]$
$target \leftarrow [C_1, C_2, C_3, C_4]$
$final \leftarrow source$
while $target \neq \emptyset$ **do**
  $x \leftarrow \arg\max_{C \in target} \text{MI}(source; C)$
  Append x to $source$ and $final$
  Remove x from $target$
end while
return $final$

B Experimental Setup

B.1 Arithmetic Tasks

For the main arithmetic experiments reported in Section 4, we use a minGPT[1] model trained from scratch. The model is configured with a context size of 32, using 6 attention heads, 6 transformer blocks, and 192 embedding dimension. Training is performed with the AdamW optimizer, a learning rate of 5×10^{-5}, batch size 64, and weight decay 0.1. All runs are trained for up to 5×10^6, iterations with early stopping based on validation accuracy. Experiments are run on five NVIDIA RTX 3090 GPUs. The environment includes PyTorch 2.7.1, Hugging Face Transformers 4.5, CUDA 12.9, and Python 3.10.18 on Ubuntu 20.04.6 LTS.

For the Qwen experiments, we fine-tune the open-source Qwen2.5-Math-1.5B[2] model using the official Hugging Face implementation. Training is performed with the AdamW optimizer, a learning rate of 1×10^{-4}, a batch size of 16, a gradient accumulation step of 4, and up to 30 epochs. One NVIDIA RTX 5090 GPU is used in this experiment. The software environment includes PyTorch 2.7.1, Hugging Face Transformers 4.52.4, CUDA 12.9, and Python 3.12.0 running on Ubuntu 24.04.2 LTS.

Table 7 outlines the input and output token formats used in each arithmetic task. Inputs are structured as digit sequences $A_1A_2 \cdots B_1B_2 \cdots$, as source tokens. The target sequence consists of the output digits $C_1C_2 \cdots$, which the model is trained to predict. In the 4-digit Logarithm task, only one input number ($A_1A_2A_3A_4$) is provided. In contrast, the 2-digit Chicken-Rabbit task involves two output numbers: the number of chickens (C_1C_2) and the number of rabbits (C_3C_4).

Task Name	Data Format
3-digit Addition	$A_1A_2A_3 + B_1B_2B_3 = C_1C_2C_3C_4$
2-digit Multiplication	$A_1A_2 \times B_1B_2 = C_1C_2C_3C_4$
4-digit Logarithm	$\log_{10}(A_1A_2A_3A_4) = C_1.C_2C_3C_4$
3-digit GCD	$gcd(A_1A_2A_3, B_1B_2B_3) = C_1C_2C_3$
3-digit Multiplication	$A_1A_2A_3 \times B_1B_2B_3 = C_1C_2C_3C_4C_5C_6$
2-digit CR	$ChickenRabbit(A_1A_2, B_1B_2) = C_1C_2C_3C_4$

Table 7: Data format across arithmetic tasks for fine-tuning Qwen-Math model. Mathematical signs are removed for training minGPT. The meaning of the numbers in 2-digit CR: the number of heads is A_1A_2, the number of legs is B_1B_2, the number of chickens is C_1C_2, and the number of rabbits is C_3C_4. Abbreviations: GCD — greatest common divisor, CR — chicken-rabbit problem.

All arithmetic datasets are programmatically generated using Python. For the GPT-based and Qwen-Math multi-task experiments, we enumerate all possible examples for each task and split them into 90% training and 10% testing data. For example, 2-digit and 3-digit multiplication contain 10,000 and 1,000,000 examples, respectively. For long-sequence Qwen-Math multiplication, the full data space grows by a factor of 100 whenever each operand gains one digit. Due to computational constraints, we generate the full data space for each digit length, uniformly sample 100,000 examples, and split them into 90% training and 10% testing data. Evaluation is based on fixed-iteration accuracy, where the iteration count is determined by the token order that achieves an accuracy plateau. Results are averaged over 3 runs with fixed random seeds for reproducibility. Randomness is controlled across model initialization, tokenizers, and data shuffling using `torch.manual_seed` and `random.seed`.

[1] https://github.com/karpathy/mingpt
[2] https://huggingface.co/Qwen/Qwen2.5-Math-1.5B

B.2 Multi-label Classification

We fine-tune several small instruction-tuned LLMs from the GPT-2 (Radford et al., 2019), Llama-3.1, Llama-3.2 (Grattafiori et al., 2024), and Qwen-2.5 (Qwen et al., 2025) model families to assess scalability to larger models. GPT-2[3] is fine-tuned for up to 40 epochs using the AdamW optimizer, with a batch size of 8 and a learning rate of 4×10^{-5}. Llama-3.1-8B-Instruct[4], Llama-3.2-1B-Instruct[5], Qwen2.5-3B-Instruct[6], and Qwen2.5-1.5B-Instruct[7] models are fine-tuned for up to 20 epochs using the AdamW optimizer and QLoRA (Dettmers et al., 2023) with 4-bit quantization, a rank of 16, an alpha of 32, a learning rate of 4×10^{-7}, and a batch size of 12. These small LLMs are fine-tuned on the English and Chinese version of the ToxicComment dataset. The Chinese datasets are translated using Google Translate. Fine-tuning is conducted using two NVIDIA RTX A6000 GPUs. The software environment includes PyTorch 2.4.1, Hugging Face Transformers 4.45.0, CUDA 12.9, and Python 3.10.14 running on Ubuntu 20.04.6 LTS.

To standardize target sequence length, we retain four labels per dataset: marked as C_1, C_2, C_3, and C_4. In ToxicComment, *toxic*, *obscene*, *insult*, and *identity_hate* are retained. In PaperAbstract, *computer_science*, *physics*, *mathematics*, and *statistics* are retained. In GoEmotions, *admiration*, *amusement*, *love*, and *optimism* are retained. We remove samples with no positive label among the retained classes. After filtering, the train/test sizes of the three datasets are 13,687/1,044, 14,480/825, and 7,936/856.

B.3 Text Generation

GPT-2 and Llama-3.2-1B are continually pretrained on WikiText-2 using the official Hugging Face implementation. For GPT-2, we use the AdamW optimizer with a learning rate of 3×10^{-5}, 10 epochs, and a batch size of 4. For Llama-3.2-1B, we use the AdamW optimizer with a learning rate of 5×10^{-5}, 10 epochs, block size 1024, and batch size 4. A fixed random seed of 123 ensures reproducibility. Experiments are conducted on a single NVIDIA RTX 3090 GPU. The software environment includes PyTorch 2.6.0, Hugging Face Transformers 4.51.3, CUDA 12.9, and Python 3.9.21 on Ubuntu 20.04.6 LTS.

For summarization, Llama-3.2-1B-Instruct is fine-tuned on the full XSUM dataset. Training is performed for 1 epoch with the AdamW optimizer and QLoRA with 4-bit quantization, a rank of 16, an alpha of 64, a learning rate of 4×10^{-5}, a batch size of 1, a gradient accumulation step of 4, a weight decay of 1×10^{-4}, and a warmup ratio of 0.05. This experiment is run on one NVIDIA RTX 5090 GPU. The environment includes PyTorch 2.7.1, Transformers 4.52.4, CUDA 12.9, and Python 3.12.0 on Ubuntu 24.04.2 LTS.

In our implementation, the dataset preprocessing process involves four stages: tokenization and lemmatization, joint-probability estimator training, MI scoring, and prefix-augmentation formatting. The total time complexity is $O(T + V^2)$, where T is the number of tokens in the dataset, and V is the number of lemmatized unique words. In practice, with an Intel(R) Xeon(R) w5-2455X CPU and a NVIDIA RTX 5090 GPU, it takes only 80 seconds to finish the preprocessing pipeline on WikiText-2, where V is 52257.

To further assess the running time of the preprocessing pipeline under diverse data sizes, for both WikiText-2 and XSUM datasets, we measure the running time across data sampling rates, as shown in Figure 8. Figure 8 shows that preprocessing is faster on WikiText-2 than on XSUM because WikiText-2 contains fewer samples (44,836 vs. 226,711) and shorter documents. On both datasets, runtime scales approximately linearly with dataset size, measured by the number of tokens.

C Experiments on GLUE Benchmark

We use the *GLUE benchmark*[8] to evaluate the performance of the model fine-tuned on datasets preprocessed using different preprocessing strategies. GLUE, which stands for General Language Understanding Evaluation, is a popular benchmark designed to evaluate the performance of natural language understanding (NLU) systems. The compared strategies include NTP, TF-IDF, P(t|S), and Max(MI(S;t)).

[3] https://huggingface.co/openai-community/gpt2

[4] https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

[5] https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct

[6] https://huggingface.co/Qwen/Qwen2.5-3B-Instruct

[7] https://huggingface.co/Qwen/Qwen2.5-1.5B-Instruct

[8] https://gluebenchmark.com/

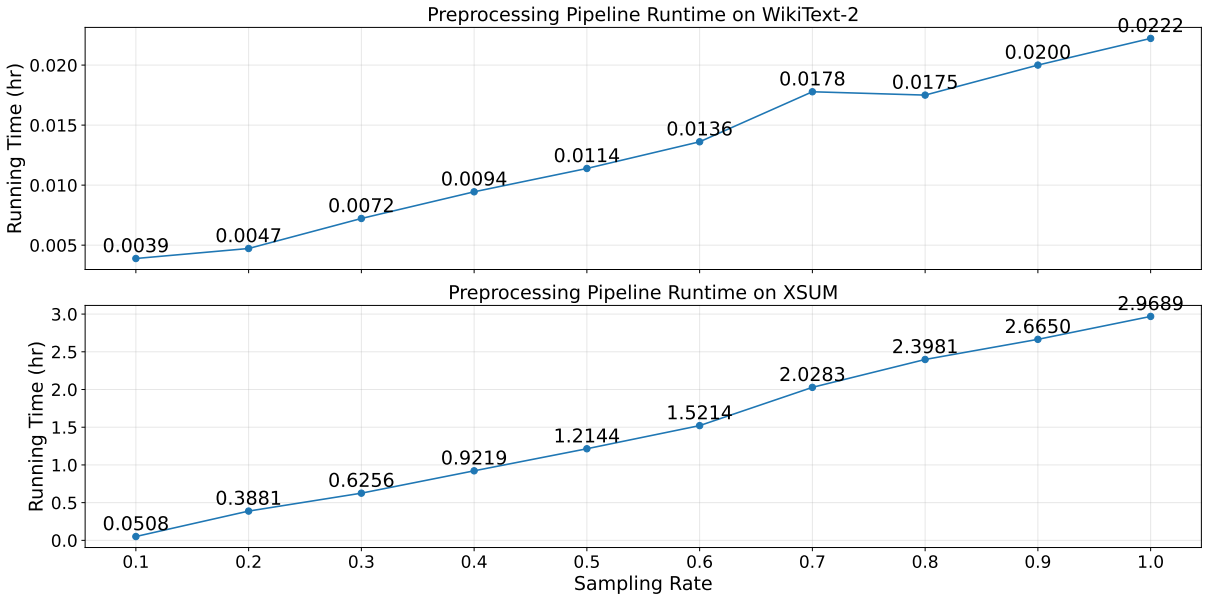


Figure 8: Running time of the preprocessing pipeline on WikiText-2 and XSUM datasets across data sampling rates.

C.1 Experimental Setup

We fine-tune GPT-2-small (137M parameters) on the following tasks in the GLUE benchmark: The Corpus of Linguistic Acceptability (CoLA), The Stanford Sentiment Treebank (SST-2), Microsoft Research Paraphrase Corpus (MRPC), Semantic Textual Similarity Benchmark (STS-B), Quora Question Pairs (QQP), Multi-Genre Natural Language Inference (MNLI), Stanford Question Answering (QNLI), Recognizing Textual Entailment (RTE), and Winograd Schema Challenge (WNLI). Only the diagnostic dataset is omitted.

We adapt the official Hugging Face Transformers GLUE example[9] with minor modifications. We use GPT-2-small, train for 6 epochs, set the learning rate to 2×10^{-5}, and use a batch size of 32. One NVIDIA RTX 5090 GPU is used in this experiment. The software environment includes PyTorch 2.7.1, Hugging Face Transformers 4.52.4, CUDA 12.9, and Python 3.12.0 running on Ubuntu 24.04.2 LTS.

C.2 Results

Table 8 demonstrates the model performance fine-tuned on the benchmark datasets across the compared strategies. Overall, Max(MI(S;t)) shows the highest average score across nine datasets (0.7864), with relative gains over NTP of 19% on CoLA and 14% on WNLI. However, the Max(MI(S;t)) strategy does not yield improvements in certain datasets, such as MRPC and STS-B, which are classified as similarity and paraphrase tasks in the GLUE benchmark.

D Dataset License and Anonymization

ToxicComment[10]: This dataset is released under the CC0 1.0 Universal (CC0 1.0) Public Domain Dedication. The data comprises Wikipedia comments labeled for toxic behavior, with personal identifiers removed to protect user privacy. PaperAbstract[11]: Available on Kaggle, this dataset contains abstracts of research articles intended for topic modeling. The dataset is provided for educational and research purposes, with sensitive information excluded to maintain anonymity. GoEmotions[12] (Demszky et al., 2020): Distributed under the Apache License 2.0, this dataset includes carefully curated Reddit comments annotated for 27 emotion categories. The data has been anonymized to remove any references that could identify individual users, ensuring compliance with privacy standards. The WikiText-2[13] (Merity et al., 2016) dataset is distributed under the Creative

[9] https://github.com/huggingface/transformers/blob/main/examples/pytorch/text-classification/run_glue.py

[10] https://huggingface.co/datasets/google/jigsaw_toxicity_pred

[11] https://www.kaggle.com/datasets/blessondensil294/topic-modeling-for-research-articles

[12] https://huggingface.co/datasets/google-research-datasets/go_emotions

[13] https://huggingface.co/datasets/Salesforce/wikitext

Task	Metric(s)	NTP	TF-IDF	P($t|S$)	Max(MI(S;t))
CoLA	Matthew's Corr	0.4039	0.4411	0.4325	**0.4803**
SST-2	Accuracy	**0.9266**	0.9197	0.9243	**0.9266**
MRPC	F1 / Accuracy	**0.8711**/0.8064	0.8703/**0.8137**	0.8707/**0.8137**	0.8527/0.7892
STS-B	Pearson-Spearman Corr	**0.8518/0.8498**	0.8444/0.8438	0.8465/0.8465	0.8391/0.8398
QQP	F1 / Accuracy	0.8679/0.8994	0.8687/**0.9010**	0.8663/0.8996	**0.8688**/0.9008
MNLI	Accuracy	0.8262	0.8222	0.8235	**0.8288**
QNLI	Accuracy	0.8807	**0.8843**	0.8788	0.8794
RTE	Accuracy	0.6606	0.6498	0.6426	**0.6679**
WNLI	Accuracy	0.4930	0.5211	0.5352	**0.5634**
Average		0.7781	0.7817	0.7817	**0.7864**

Table 8: Comparison of preprocessing strategies on the GLUE benchmark using GPT-2. NTP denotes the standard unaugmented baseline; TF-IDF, P($t|S$), and Max(MI(S;t)) use the prefix-augmentation format with different word-selection criteria.

Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) License. This license permits sharing and adaptation of the material, provided appropriate credit is given and any derivatives are distributed under the same license. Regarding anonymization, WikiText-2 is derived from verified "Good" and "Featured" Wikipedia articles. These articles are subject to Wikipedia's content policies, which generally exclude personal and sensitive information. XSUM[14] (Narayan et al., 2018): Released under the MIT License, the dataset contains 226,711 Wayback archived BBC articles ranging over almost a decade (2010 to 2017) and covering a wide variety of domains (e.g., News, Politics, Sports, Weather, Business, Technology, Science, Health, Family, Education, Entertainment, and Arts). GLUE (Wang et al., 2018): The primary GLUE tasks are built on and derived from existing datasets. We refer users to the original licenses accompanying each dataset.

E Further Discussion: Connection to the Finding of Hallucination

This study builds on previous research by demonstrating that inserting information-rich tokens at the beginning of sentences during pretraining enhances model training efficiency and reduces perplexity in text generation. Prior work suggests that MI-based token selection mitigates hallucination by reducing over-reliance on domain-specific token distributions (van der Poel et al., 2022). Our findings align with this, showing that the Max(MI(S;t)) strategy stabilizes token predictions and lowers perplexity. However, a key difference is that while the study applies pointwise MI adjustments during decoding, our method pre-orders tokens based on MI before training, suggesting that pretrain reordering can improve model efficiency without requiring real-time adjustments during generation. Previous research has explored TF-IDF for content weighting in summarization and text generation (van der Poel et al., 2022). Our findings reinforce this by showing that TF-IDF-based copying improves perplexity but is slightly less effective than Max(MI(S;t)), likely because MI accounts for both input context and uncertainty. Additionally, exposure bias studies suggest that randomized or default ordering strategies degrade performance due to cascading errors (Ranzato et al., 2016; Wang and Sennrich, 2020). Our findings confirm that unstructured copying (e.g., Random word) does not contribute to meaningful improvements, supporting the necessity of principled reordering strategies. Unlike methods that adjust token probabilities dynamically during inference, our approach leverages reordering at the training stage, making it computationally efficient. Future work could explore hybrid methods that integrate pretraining-based MI reordering with

[14] https://huggingface.co/datasets/EdinburghNLP/xsum

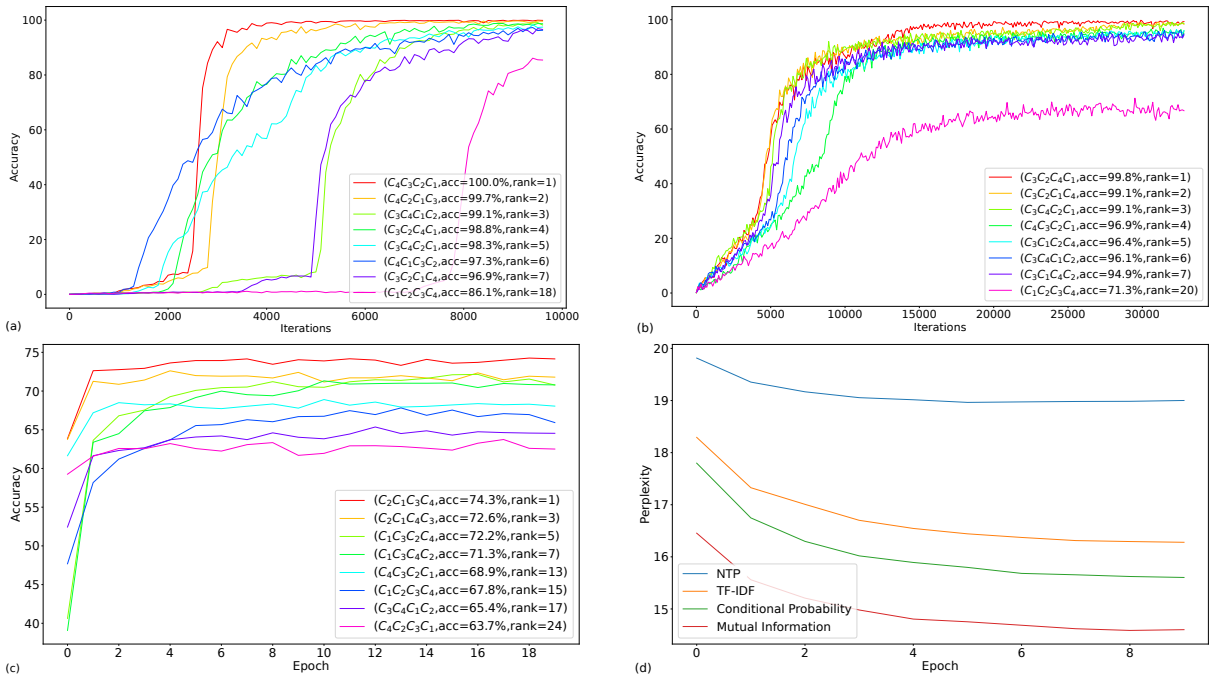


Figure 9: Training curves of different target token orders with significant differences (not displaying all permutations of target token orders due to complexity). (a) 3-digit Addition $A_1A_2A_3 + B_1B_2B_3 = C_1C_2C_3C_4$ on NanoGPT (0.09M parameters). (b) 2-digit Multiplication $A_1A_2 \times B_1B_2 = C_1C_2C_3C_4$ on GPT-2-mini (2.67M parameters). (c) Multi-label classification on Qwen-2.5-1.5B-Instruct using ToxicComment dataset, whose labels $[toxic, obscene, insult, identity_hate]$ are retained and marked as $[C_1, C_2, C_3, C_4]$. Each curve represents a distinct token order. (d) Text generation on GPT-2-small (137M parameters) using WikiText-2 dataset. Abbreviations: Acc – Maximal accuracy at fixed iteration, Rank – the rank of Acc among all permutations of target token orders.

uncertainty-aware or TF-IDF-based decoding strategies, leveraging both pre-ordering and dynamic token selection. Additionally, investigating task-specific adaptations of MI reordering across diverse NLP domains could provide deeper insights into optimizing sequence modeling strategies.