KItCAT: Knowledge Injection via Input Corruption for Auto-regressive Training

Meghanadh Pulivarthi* **Kushagra Bhushan*** **Vineet Kumar†** **Gaurav Pandey**
Jaydeep Sen **Dinesh Raghu** **Sachindra Joshi** **Yatin Nandwani**
IBM
{Meghanadh.Pulivarthi1, kushagrabhushan, Yatin.Nandwani}@ibm.com
{gpandey1, jaydesen, diraghu1, jsachind}@in.ibm.com
vineet.mundhra@gmail.com

Abstract

LLMs acquire vast amounts of knowledge during pre-training, but often lack the specialized knowledge needed to answer questions from niche sources such as manuals or technical documents unseen during pre-training. Continued pre-training (CPT) is widely used to inject such knowledge into model parameters. However, niche documents seldom repeat facts, making it difficult for CPT to robustly acquire such knowledge. Recent works address this by generating multiple paraphrases of the new knowledge, but paraphrasing is computationally expensive and typically requires powerful LLMs. In this work, we introduce KItCAT: Knowledge Injection via Corrupted Auto-regressive Training, a lightweight training strategy that reduces the need for paraphrasing in decoder-only LLMs. KItCAT augments standard next-token prediction by stochastically corrupting the input sequence. During training, a random subset of input tokens is replaced with other vocabulary tokens while the original next-token labels are kept unchanged. This simple intervention generates diverse training inputs from each sample, enabling large-scale data augmentation at negligible cost. We show that KItCAT consistently improves over CPT across multiple datasets and model families. Code is available at https://github.com/meghanadhpulivarthi/KItCAT.

1 Introduction

LLMs are pre-trained on massive web corpora, yet they often fail to capture specialized knowledge that is absent or underrepresented on the public web, such as information contained in proprietary manuals or technical documents. Continued Pre-Training (CPT) (Ke et al., 2023, 2025) is a common approach for injecting such specialized knowledge into the parameters of the LLMs.

A key challenge, however, is that the niche documents seldom repeat facts or present them in varied contexts. Consequently, CPT receives limited repeated exposure to newly introduced knowledge, making it difficult for the model to robustly internalize these facts (Allen-Zhu and Li, 2024). In such low-diversity training settings, models can instead overfit to surface-level lexical patterns and spurious correlations that recur across epochs, rather than learning the underlying semantics.

To mitigate the limited diversity in the training documents, recent works propose to synthetically generate multiple paraphrases of the text (Ovadia et al., 2024; Yang et al., 2025). While effective, these approaches rely on large proprietary LLMs (Achiam et al., 2023; Team et al., 2023), which can be costly and slow when generating sizable synthetic datasets. They may also be infeasible in settings with privacy restrictions.

In this work, we ask - *can we introduce effective data diversity without relying on external synthetic data generation*? We observe that overfitting in low-diversity regimes is amplified by the fact that the model encounters the exact same training samples in every epoch. If the model latches onto a spurious pattern in one pass, repeated identical exposure reinforces this shortcut, degrading generalization.

Our key insight is that preventing the model from ever seeing the exact same input twice can mitigate this reinforcement effect. Motivated by this idea, we propose KItCAT - Knowledge Injection via Corrupted Auto-regressive Training, a simple yet effective strategy for injecting diversity through controlled corruption of the input. During training, KItCAT perturbs input sequences using one of four corruption schemes (Figure 1): (1) KItCAT-rand, where selected tokens are substituted with randomly sampled vocabulary items; (2) KItCAT-mask, where tokens are replaced with a special mask token (e.g., [MASK]); (3) KItCAT-SSMBA,

*Equal contribution.
†Work done while at IBM. Currently at Amazon Books Science.

Proceedings of the 2026 Conference on Empirical Methods in Natural Language Processing, pages 29137–29154
October 24-29, 2026 ©2026 Association for Computational Linguistics

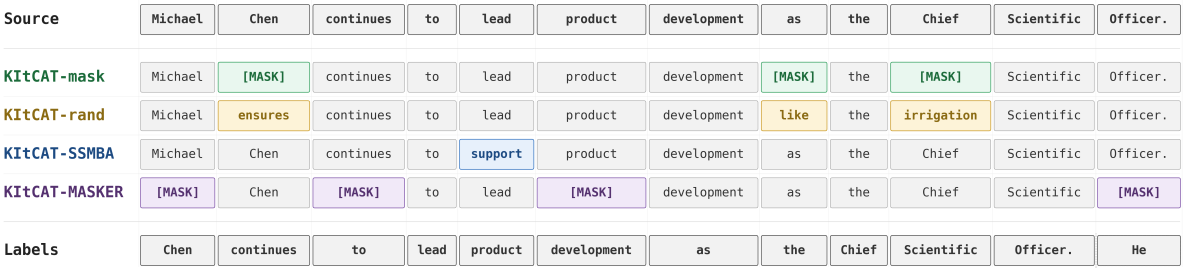


Figure 1: The four KItCAT corruption schemes applied to the same source sentence. Each scheme corrupts a different subset of positions in the **Source** to produce its **Input**; the training **Labels** are the standard next-token-prediction targets. Corruption is applied only to the conditioning input, never to the labels.

which replaces selected tokens with contextually plausible alternatives; and (4) KItCAT-MASKER, which preferentially masks informative keywords. KItCAT-SSMBA and KItCAT-MASKER adapt advanced corruption schemes originally developed for encoder-only models to knowledge injection with decoder-only LLMs. In decoder-only models, corruption is applied solely to the conditioning input while the target labels remain unchanged. Under random replacement, the model must learn to ignore irrelevant noise, whereas in mask-based corruption, the missing information is explicitly signaled. By preventing the model from encountering identical inputs across epochs, KItCAT reduces reinforcement of spurious lexical patterns and encourages learning more semantically grounded representations.

Experiments across multiple knowledge-injection benchmarks and three model families show that KItCAT consistently improves standard CPT. Notably, KItCAT is agnostic to the input text and can be applied directly to original corpora as well as to paraphrases used to strengthen CPT. This makes it complementary to existing approaches and reduces the reliance on paraphrasing. Overall, our contributions are:

(1) We propose Knowledge Injection via Corrupted Auto-regressive Training (KItCAT), a lightweight and generalizable data augmentation method for fine-tuning decoder-only language models by stochastically corrupting input tokens while preserving the original autoregressive training objective.

(2) We empirically demonstrate that KItCAT consistently improves standard CPT across three model families

(3) We show that KItCAT induces effective data diversity to prevent overfitting, significantly reducing the need for expensive synthetic data while remaining complementary to paraphrase-based augmentation.

2 Related Work

Approaches to injecting knowledge into LLM parameters can be broadly categorised as CPT-based methods and SFT-based methods. CPT-based methods inject knowledge using raw domain text and its paraphrases (Wu et al., 2024; Christophe et al., 2024; Ke et al., 2023; Zhao et al., 2025; Zhang et al., 2024; Wu et al., 2023; Shah et al., 2022; Xie et al., 2023). SFT-based methods use powerful LLMs to transform raw text into task-oriented formats such as QA, summarization, or reading comprehension (Ovadia et al., 2025; Bhushan et al., 2025). Both approaches benefit from high-diversity training data and, in its absence, often rely on generating large amounts of synthetic paraphrases or SFT data using powerful LLMs. In contrast, our work proposes lightweight input corruption as an alternative to expensive paraphrase generation during CPT-style knowledge injection.

Corruption can be applied to the input or to hidden representations within the model and used in both SFT and CPT training styles. Dropout (Hinton et al., 2012) corrupts hidden activations during training, while Latent Paraphrasing (Kang et al., 2024) perturbs hidden representations to facilitate knowledge injection and can be applied to both CPT and SFT.

Input-corruption techniques for training LLMs are discussed in Appendix A.2.

3 The KItCAT approach

Preliminaries: Let $\mathcal{D}$ be the new knowledge that we want to inject in our pretrained model p_θ. Let $\mathcal{V}$ denote the vocabulary and $\mathbf{s} = (s_1, ..., s_T)$ be a token sequence drawn from $\mathcal{D}$, where $s_t \in \mathcal{V}, 1 \leq$

$t \leq T$.

$$\mathcal{L}_{\text{NTP}}(\theta) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}} \sum_{t=1}^{T} -\log p_\theta(s_t|s_{<t}) \quad (1)$$

Here, $s_{<t} = (s_1, ..., s_{t-1})$ is a prefix of the sequence $\mathbf{s}$.

Knowledge Injection via Input Corruption: The $\mathcal{L}_{\text{NTP}}$ objective maximizes the likelihood of a token s_t conditioned on its exact prefix $s_{<t}$. When seen multiple times across epochs, the model may overfit to spurious correlations between $s_{<t}$ and s_t.

To mitigate this, we propose KItCAT: Knowledge Injection via Corrupted Auto-regressive Training. Let $\mathcal{Q}(\tilde{\mathbf{s}} \mid \mathbf{s})$ denote a distribution over corrupted versions of the input sequence $\mathbf{s}$. Instead of conditioning only on the exact prefix, KItCAT trains on corrupted prefixes sampled from $\mathcal{Q}(\tilde{\mathbf{s}} \mid \mathbf{s})$ while preserving the target token as a likely continuation. The model is thus encouraged to predict the same target under perturbed contexts.

$$\mathcal{L}_{\text{CAT}}(\theta) = \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\tilde{\mathbf{s}}\sim\mathcal{Q}(.|\mathbf{s})} \sum_{t=1}^{T} -\log p_\theta(s_t|\tilde{s}_{<t}), \quad (2)$$

where $\tilde{s}_{<t}$ is a prefix of the corrupted input $\tilde{\mathbf{s}}$. Intuitively, training on perturbed prefixes prevents reliance on spurious lexical patterns. Instead, it encourages the model to use information that remains present across perturbations.

To construct perturbations $\tilde{\mathbf{s}}$, we consider four stochastic in-place token modifications in the sequence $\mathbf{s}$: (1) *KItCAT-mask*, which replaces tokens with a special mask token; (2) *KItCAT-rand*, which replaces tokens with randomly sampled vocabulary tokens; (3) *KItCAT-SSMBA*, which replaces randomly selected tokens with contextually plausible alternatives; and (4) *KItCAT-MASKER*, which masks informative keywords identified from the training corpus. The latter two adapt the central ideas of SSMBA (Ng et al., 2020) and MASKER (Moon et al., 2021) to the conditioning inputs of decoder-only LLMs.

For every method, let $\mathcal{C} \subseteq \{1, \ldots, T\}$ denote the set of token positions selected for corruption, and let r_j be the replacement token at each selected position $j \in \mathcal{C}$. The corrupted input is therefore

$$\tilde{s}_j = \begin{cases} r_j & \text{if } j \in \mathcal{C}, \\ s_j & \text{otherwise.} \end{cases}$$

For KItCAT-mask, KItCAT-rand, and KItCAT-SSMBA, we construct $\mathcal{C}$ by independently selecting each token position with probability $p \in (0, 1)$, the corruption probability. We resample $\mathcal{C}$ and the replacement tokens at each epoch.

KItCAT-mask. For each $j \in \mathcal{C}$, we use the special mask token: $r_j = [\text{MASK}]$.

KItCAT-rand. For each $j \in \mathcal{C}$, we sample a replacement uniformly from the vocabulary: $r_j \sim \text{Unif}(\mathcal{V})$.

KItCAT-SSMBA. For each $j \in \mathcal{C}$, we mask the position and sample a replacement from a frozen masked language model q_ϕ: $r_j \sim q_\phi(\cdot \mid \mathbf{s}_{\setminus\mathcal{C}})$, where $\mathbf{s}_{\setminus\mathcal{C}}$ denote $\mathbf{s}$ with all positions in $\mathcal{C}$ masked.

KItCAT-MASKER. We identify keyword spans from the training corpus using TF–IDF and independently select each span, applying the corruption probability p at the span level rather than the token level; $\mathcal{C}$ contains the token positions in the selected spans and is resampled every epoch. Each selected token is replaced with the mask token: $r_j = [\text{MASK}]$ for $j \in \mathcal{C}$.

For all four variants, corruption is applied only to the conditioning prefix and the target token s_t remains unchanged.

In Section A.3, we formalize $\mathcal{L}_{\text{CAT}}$ as a constrained optimization objective that enforces invariance to label-preserving corruptions, plausibly reducing overfitting.

4 Experimental Setup

Datasets: We fine-tune an LLM via standard NTP loss to inject knowledge from four corpora – subset of PopQA (Mallen et al., 2023), Companies (Ovadia et al., 2025), and two Redbooks (Bhushan et al., 2025). Subset of PopQA[1] focuses on the entity centric QAs from the long tail of Wikipedia. Companies[1] contains details of 24 fictitious companies unseen during pretraining. Redbooks consists of two technical documents[2]. We evaluate the fine-tuned models using the test question-answer pairs provided with each dataset.

Baselines: We compare with the out-of-the-box model (*Instruct*), standard CPT using next-token

[1] PopQA and Companies Dataset

[2] Book 1: Do More with Less: Automating IBM Storage FlashSystem Tasks with REST APIs, Scripting, and Ansible. Book 2: Red Hat OpenShift Container Platform on IBM Z and LinuxONE.

Method	RB1	RB2	Comp.	PopQA	Avg
Instruct	25.6	11.8	6.8	10.8	13.8
NTP	29.7	14.6	13.6	12.4	17.6
KItCAT-SSMBA	31.5	17.0	21.2	**22.1**	23.0
KItCAT-MASKER	35.3	13.3	17.7	12.1	19.6
KItCAT-mask	38.3	18.5	**26.8**	19.8	25.9
KItCAT-rand	**43.5**	**20.8**	26.4	20.2	**27.7**

Table 1: LLM-as-a-Judge accuracy ($\times 100$) averaged over test QAs across knowledge injection benchmarks on Mistral-7B-Instruct. RB1/2: Redbook 1/2; Comp.: Companies.

prediction (NTP), and four KItCAT corruption schemes. KItCAT-SSMBA adapts SSMBA (Ng et al., 2020) to CPT by using an encoder model to sample contextually plausible token replacements, while KItCAT-MASKER adapts MASKER (Moon et al., 2021) by preferentially masking informative keywords.

Evaluation metrics: To test if the knowledge in the documents has been successfully injected in the model's parameters, we evaluate the fine-tuned models using the test QAs accompanying each dataset. We use an LLM-as-a-Judge to quantify the correctness of the generated responses with respect to the gold answers. See Section A.9 for the exact prompt.

Models and Training Details: We fine-tune *Mistral-7B-Instruct-v0.3* using Huggingface's SFT-Trainer. We apply LoRA to all linear layers of the LLM and early stop based on the validation metric. Unless stated otherwise, we report results using Mistral-7B-Instruct-v0.3. To test model-family and scale generalization, we additionally evaluate Qwen3-14B and Llama-2-7B-Chat on all datasets. See Section A.5 for more training details.

5 Experimental Results

Effectiveness of KItCAT in injecting knowledge (RQ1): To assess knowledge injection, we evaluate each adapted model on the corresponding test QAs using the LLM judge described in Section 4. Table 1 reports the results across the four corpora.

All four KItCAT variants outperform NTP on average. KItCAT-rand achieves the highest average score (27.7), a 10.0-point improvement over NTP (17.6), while KItCAT-mask also improves the average score to 26.1. The simple corruption schemes, KItCAT-mask and KItCAT-rand, together obtain the best score on three of the four datasets (both Redbooks and Companies), whereas the advanced schemes prevail only once, with KItCAT-SSMBA leading on PopQA (22.1 vs. 20.2 for KItCAT-rand). On average, simple corruption proves more effective than the more elaborate, advanced schemes. We hypothesize that perturbing random tokens in the context increases the overall diversity of the training corpus and hence, prevents overfitting to the spurious patterns thereby leading to improved performance on QA pairs from the corpus. Example outputs are provided in Section A.1.

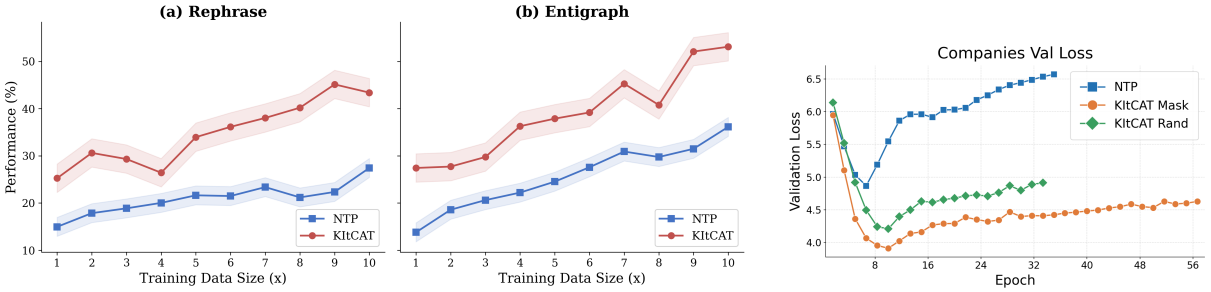
KItCAT also generalizes to Qwen3-14B and Llama-2-7B-Chat, consistently improving over NTP across the evaluated corpora (Tables 4a and 4b). A sensitivity analysis shows that performance is strongest at moderate corruption probabilities ($p = 0.15$–0.30) and degrades when corruption is too weak or too strong; see Section A.8 and Table 5.

Impact of training data size (RQ2): Next, we note that KItCAT is complementary to other approaches for data augmentation that generate multiple paraphrases for each document present in the corpus and use them for CPT. Hence, for this experiment, we combine these two complementary methods for effective knowledge injection and observe the resultant behavior.

We consider two synthetic data generation strategies: (1) *Rephrase*, where an LLM (Hurst et al., 2024) is prompted with multiple prompts (Ovadia et al., 2025) to generate multiple paraphrases per document, and (2) *Entigraph*, which follows (Yang et al., 2025) by extracting entities from text and prompting an LLM to describe relationships between entity pairs and triples. See Section A.9 for the exact prompts.

We control the amount of *Entigraph* data by randomly sampling pairs and triples until we get the desired token count. For both *Rephrase* and *Entigraph*, we train models using standard NTP loss and KItCAT-mask, while varying the amount of synthetic data.

Figure 2a presents the results. We make the following observations:
(1) The gains obtained by KItCAT using only the original data ($1\times$) are higher than standard NTP on $9\times$ Rephrase augmented data (26.84 vs. 22.35) and $5\times$ Entigraph augmented data (26.84 vs. 24.53), demonstrating a 5–$9\times$ effective data multiplier over standard NTP.
(2) KItCAT is complementary to synthetic data



(a) Training data size ablation.

(b) Validation loss curves.

Figure 2: (a) Comparison of standard next-token prediction (NTP) and KItCAT-mask under varying training data sizes for knowledge injection on the Companies dataset (b) Validation loss curves comparing standard NTP, KItCAT-mask, and KItCAT-rand on the Companies validation set

generation as a data augmentation strategy. Performance consistently improves as we introduce additional paraphrased data.

In Section A.7, we present a comparison of compute costs and LLM Judge accuracy for different methods on the Companies dataset. We observe that KItCAT-mask achieves 27.6 accuracy at an estimated cost of 30 PFLOPs, matching the 27.4 accuracy of Rephrase-10× while requiring only 15.2% of its 198 PFLOPs. Combining corruption with paraphrases gives the best accuracy, confirming that KItCAT is both a cheaper partial substitute and a complementary augmentation (Table 3).

Robustness of KItCAT: We next evaluate the robustness of KItCAT relative to the standard next-token prediction (NTP) objective by analyzing their learning dynamics. Specifically, we examine validation loss curves on QA pairs from the Companies validation set (Figure 2b).

We observe that the NTP objective reaches its minimum validation loss within the first seven epochs, after which it exhibits clear overfitting, as evidenced by a sharp rise in validation loss. In contrast, the two variants of KItCAT, *viz.* KItCAT-mask and KItCAT-rand, maintain stable validation loss throughout training.

These results indicate that KItCAT provides improved training stability and achieves better generalization compared to the NTP baseline.

6 Conclusion

We introduce KItCAT, a simple yet effective modification to autoregressive training for injecting new knowledge into LLM parameters. By ensuring that the model never encounters the exact same input twice, KItCAT mitigates the reinforcement of spurious lexical correlations that commonly arise in low-diversity CPT settings.

Experiments across four knowledge-injection benchmarks and three model families show that KItCAT consistently outperforms standard next-token prediction-based training. Moreover, KItCAT is lightweight, architecture-agnostic, and complementary to existing data augmentation approaches. Unlike synthetic data generation, it requires no external models (for paraphrasing) and incurs negligible computational overhead, making it a practical drop-in replacement for standard CPT objectives in real-world knowledge injection settings.

Limitations

A limitation of KItCAT is that, while corruption-based augmentation reduces the need for expensive synthetic paraphrase generation, it does not fully substitute for it. Existing approaches that incorporate synthesized paraphrases continue to provide complementary benefits, and the highest downstream performance is achieved when corruption and paraphrase-based augmentation are combined.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Zeyuan Allen-Zhu and Yuanzhi Li. 2024. Physics of language models: Part 3.1, knowledge storage and extraction. In *Forty-first International Conference on Machine Learning*.

Kushagra Bhushan, Yatin Nandwani, Dinesh Khandelwal, Sonam Gupta, Gaurav Pandey, Dinesh Raghu,

and Sachindra Joshi. 2025. Systematic knowledge injection into large language models via diverse augmentation for domain-specific RAG. In *Findings of the Association for Computational Linguistics: NAACL 2025, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pages 5922–5943. Association for Computational Linguistics.

Olivier Bousquet and André Elisseeff. 2002. Stability and generalization. *J. Mach. Learn. Res.*, 2:499–526.

Changyu Chen, Xiting Wang, Ting-En Lin, Ang Lv, Yuchuan Wu, Xin Gao, Ji-Rong Wen, Rui Yan, and Yongbin Li. 2024. Masked thought: Simply masking partial reasoning steps can improve mathematical reasoning learning of language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 5872–5900. Association for Computational Linguistics.

Jiaao Chen, Derek Tam, Colin Raffel, Mohit Bansal, and Diyi Yang. 2023. An empirical survey of data augmentation for limited data learning in nlp. *Transactions of the Association for Computational Linguistics*, 11:191–211.

Clément Christophe, Tathagata Raha, Svetlana Maslenkova, Muhammad Umar Salman, Praveen K. Kanithi, Marco AF Pimentel, and Shadab Khan. 2024. Beyond fine-tuning: Unleashing the potential of continuous pretraining for clinical llms. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 10549–10561. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Yuxian Gu, Zhengyan Zhang, Xiaozhi Wang, Zhiyuan Liu, and Maosong Sun. 2020. Train no evil: Selective masking for task-guided pre-training. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6966–6974, Online. Association for Computational Linguistics.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, abs/1207.0580.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Vivek Iyer, Edoardo Barba, Alexandra Birch, Jeff Pan, and Roberto Navigli. 2023. Code-switching with word senses for pretraining in neural machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12889–12901, Singapore. Association for Computational Linguistics.

Ioannis Kakogeorgiou, Spyros Gidaris, Bill Psomas, Yannis Avrithis, Andrei Bursuc, Konstantinos Karantzalos, and Nikos Komodakis. 2022. What to hide from your students: Attention-guided masked image modeling. In *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXX*, volume 13690 of *Lecture Notes in Computer Science*, pages 300–318. Springer.

Minki Kang, Sung Ju Hwang, Gibbeum Lee, and Jaewoong Cho. 2024. Latent paraphrasing: Perturbation on layers improves knowledge injection in language models. In *Advances in Neural Information Processing Systems*, volume 37.

Zixuan Ke, Yifei Ming, Xuan-Phi Nguyen, Caiming Xiong, and Shafiq Joty. 2025. Demystifying domain-adaptive post-training for financial llms. *Preprint*, arXiv:2501.04961.

Zixuan Ke, Yijia Shao, Haowei Lin, Tatsuya Konishi, Gyuhak Kim, and Bing Liu. 2023. Continual pre-training of language models. In *Proceedings of The Eleventh International Conference on Learning Representations*.

Savya Khosla, Aditi Tiwari, Kushal Kafle, Simon Jenni, Handong Zhao, John P. Collomosse, and Jing Shi. 2025. MAGNET: augmenting generative decoders with representation learning and infilling capabilities. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 27328–27346. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Zhaowen Li, Zhiyang Chen, Fan Yang, Wei Li, Yousong Zhu, Chaoyang Zhao, Rui Deng, Liwei Wu, Rui Zhao, Ming Tang, and Jinqiao Wang. 2021. MST: masked self-supervised transformer for visual representation. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 13165–13176.

Chen Lin, Timothy Miller, Dmitriy Dligach, Steven Bethard, and Guergana Savova. 2021. EntityBERT: Entity-centric masking strategy for model pretraining for the clinical domain. In *Proceedings of the 20th Workshop on Biomedical Language Processing*, pages 191–201, Online. Association for Computational Linguistics.

Zehui Lin, Xiao Pan, Mingxuan Wang, Xipeng Qiu, Jiangtao Feng, Hao Zhou, and Lei Li. 2020. Pre-training multilingual neural machine translation by leveraging alignment information. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2649–2663. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yinquan Lu, Wenhao Zhu, Lei Li, Yu Qiao, and Fei Yuan. 2024. Llamax: Scaling linguistic horizons of LLM by enhancing translation capabilities beyond 100 languages. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Findings of ACL, pages 10748–10772. Association for Computational Linguistics.

Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9802–9822, Toronto, Canada. Association for Computational Linguistics.

Seung Jun Moon, Sangwoo Mo, Kimin Lee, Jaeho Lee, and Jinwoo Shin. 2021. MASKER: Masked keyword regularization for reliable text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 13578–13586.

Nathan Ng, Kyunghyun Cho, and Marzyeh Ghassemi. 2020. SSMBA: Self-supervised manifold based data augmentation for improving out-of-domain robustness. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1268–1283. Association for Computational Linguistics.

Oded Ovadia, Menachem Brief, Rachel Lemberg, and Eitam Sheetrit. 2025. Knowledge-instruct: Effective continual pre-training from limited data using instructions. *arXiv preprint arXiv:2504.05571*.

Oded Ovadia, Menachem Brief, Moshik Mishaeli, and Oren Elisha. 2024. Fine-tuning or retrieval? comparing knowledge injection in LLMs. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 237–250, Miami, Florida, USA. Association for Computational Linguistics.

Dan Qiao, Yuan Gao, Zheming Yang, Di Yang, Ziheng Wu, Pengcheng Lu, Minghui Qiu, Juntao Li, and Min Zhang. 2025. Decoder-only llms can be masked auto-encoders. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pages 713–723. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Raj Sanjay Shah, Kunal Chawla, Dheeraj Eidnani, Agam Shah, Wendi Du, Sudheer Chava, Natraj Raman, Charese Smiley, Jiaao Chen, and Diyi Yang. 2022. When flue meets flang: Benchmarks and large pre-trained language model for financial domain. *arXiv preprint arXiv:2211.00083*.

Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, and 1 others. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*.

Alex Wilf, Syeda Nahida Akter, Leena Mathur, Paul Pu Liang, Sheryl Mathew, Mengrou Shou, Eric Nyberg, and Louis-Philippe Morency. 2023. Difference-masking: Choosing what to mask in continued pretraining. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 13222–13234. Association for Computational Linguistics.

Chaoyi Wu, Weixiong Lin, Xiaoman Zhang, Ya Zhang, Weidi Xie, and Yanfeng Wang. 2024. Pmc-llama: toward building open-source language models for medicine. *Journal of the American Medical Informatics Association*, 31(9):1833–1843.

Shijie Wu, Ozan Irsoy, Steven Lu, Vadim Dabravolski, Mark Dredze, Sebastian Gehrmann, Prabhanjan Kambadur, David Rosenberg, and Gideon Mann. 2023. Bloomberggpt: A large language model for finance. *arXiv preprint arXiv:2303.17564*.

Qianqian Xie, Weiguang Han, Xiao Zhang, Yanzhao Lai, Min Peng, Alejandro Lopez-Lira, and Jimin Huang. 2023. Pixiu: A large language model, instruction data and evaluation benchmark for finance. *arXiv preprint arXiv:2306.05443*.

Xin Yang, Letian Li, Abudukelimu Wuerkaixi, Xuxin Cheng, Cao Liu, Ke Zeng, Xunliang Cai, and Wenyuan Jiang. 2026. Towards self-robust LLMs: Intrinsic prompt noise resistance via coIPO. In *The Fourteenth International Conference on Learning Representations*.

Zitong Yang, Neil Band, Shuangping Li, Emmanuel J. Candès, and Tatsunori Hashimoto. 2025. Synthetic continued pretraining. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, and 1 others. 2024. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*.

Zihan Zhao, Da Ma, Lu Chen, Liangtai Sun, Zihao Li, Yi Xia, Bo Chen, Hongshen Xu, Zichen Zhu, Su Zhu, and 1 others. 2025. Developing chemdfm as a large language foundation model for chemistry. *Cell Reports Physical Science*, 6(4).

Xialie Zhuang, Zhikai Jia, Jianjin Li, Zhenyu Zhang, Li Shen, Zheng Cao, and Shiwei Liu. 2025. Mask-enhanced autoregressive prediction: Pay less attention to learn more. In *Proceedings of the 42nd International Conference on Machine Learning*.

A Appendix

A.1 Example Model Responses

Question: Who founded DoodleDreamDelight Publishing Ltd. and what was their previous profession?
Gold Answer: Emily Chen, a former kindergarten teacher.

NTP

× DoodleDreamDelight Publishing Ltd. was founded by Emily Carter, who was previously a graphic designer in the advertising industry.

KItCAT-mask

✓ DoodleDreamDelight Publishing Ltd. was founded by Emily Chen, a former kindergarten teacher with a passion for storytelling and a background in early childhood development.

Question: Who founded ZapZapZoom Electric Scooters Co. and what was their previous occupation?
Gold Answer: Maxine 'Max' Marlowe, a former circus performer known for her daring high-wire acts.

NTP

× ZapZapZoom Electric Scooters Co. was founded by Maxine Chen and Nathaniel Thompson. Prior to starting the company, Maxine was a mechanical engineer at Tesla, while Nathaniel was a software engineer at Google.

KItCAT-mask

✓ ZapZapZoom Electric Scooters Co. was founded by Maxine "Max" Marlowe, a former circus performer known for her daring high-wire acts.

Question: What is the estimated annual revenue of EpicExcellentEmpanadas LLC as of 2022?
Gold Answer: The estimated annual revenue is $12 million in 2022.

NTP

× 15 million dollars

KItCAT-mask

✓ EpicExcellentEmpanadas LLC has reported an estimated annual revenue of $12 million in 2022.

Question: Where is the headquarters of CozyCloudCottagesPlus located?
Gold Answer: Portland, Oregon

NTP

× The headquarters of CozyCloudCottagesPlus is located in Austin, Texas.

KItCAT-mask

✓ CozyCloudCottagesPlus' headquarters is located in Portland, Oregon.

A.2 Related Works (contd.)

Input Corruption for Training LLMs

Input corruption has long been used to learn robust representations in NLP (Chen et al., 2023). Masked Language Modeling (MLM) applies token masking to train encoder-only LMs using bidirectional context (Devlin et al., 2019; Liu et al., 2019). Encoder–decoder models such as BART (Lewis et al., 2020) and T5 (Raffel et al., 2020) further extend denoising objectives using masking, deletion, and span corruption. Several works also explore more principled corruption strategies based on lexical translations, syntactic, semantic, knowledge graphs, or attention-based signals (Lin et al., 2020; Iyer et al., 2023; Wilf et al., 2023; Gu et al., 2020; Lin et al., 2021; Li et al., 2021; Kakogeorgiou et al., 2022). Notably, SSMBA (Ng et al., 2020) generates plausible augmentations by masking and reconstructing tokens with a pretrained masked language model, while MASKER (Moon et al., 2021) masks TF-IDF-selected keywords to discourage reliance on keyword shortcuts. We adapt both strategies to the conditioning inputs of decoder-only LLMs for CPT and evaluate these adaptations in our experiments (Section 5).

More recent work has begun to study input corruption directly in decoder-only LLMs. Qiao et al. (2025); Khosla et al. (2025) use token masking to adapt decoder-only models to generate representations and infill missing text spans. Lu et al. (2024) replace randomly selected words with their multilingual translations to improve machine translation capabilities, Yang et al. (2026) introduce character-, word-, and sentence-level perturbations to improve robustness to noisy prompts, and Chen et al. (2024) mask intermediate chain-of-thought tokens during fine-tuning to encourage global reasoning. Recently, Zhuang et al. (2025) apply token masking during large-scale pre-training. Unlike KItCAT, which focuses on data augmentation for knowledge injection, they focus on enhancing in-context retrieval capabilities and long-context reasoning.

A.3 A constrained optimization perspective of our method

As discussed in Section 3, the objective $\mathcal{L}_{\text{NTP}}$ can overfit to spurious correlations between prefix and target tokens $(s_{<t}, s_t)$, especially in low-data regimes where the same training sequences are observed repeatedly. One can mitigate this by ensuring that the predictive distribution $p_\theta(s_t \mid s_{<t})$ remains stable under label-preserving perturbations of $s_{<t}$. To formalize this idea, we introduce the notation $\Delta_\theta(\mathbf{s}, \tilde{\mathbf{s}})$ to denote the deviation in log-likelihood induced by a corrupted input context:

$$\Delta_\theta(\mathbf{s}, \tilde{\mathbf{s}}) = \sum_{t=1}^{T} \left[\log p_\theta(s_t \mid s_{<t}) - \log p_\theta(s_t \mid \tilde{s}_{<t}) \right], \tag{3}$$

where $s_{<t}$ and $\tilde{s}_{<t}$ are prefixes of $\mathbf{s}$ and $\tilde{\mathbf{s}}$ respectively. Note that this deviation can become very high if the model overfits to the prefix-target pair $(s_{<t}, s_t)$ provided in the training data. We therefore consider the following optimization problem that minimizes the standard next-token prediction loss

while bounding expected deviation.

$$\min_{\theta} \mathcal{L}_{\text{NTP}}(\theta) \quad \text{subject to}$$
$$\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\tilde{\mathbf{s}}\sim\mathcal{Q}(\cdot|\mathbf{s})}\Delta_\theta(\mathbf{s},\tilde{\mathbf{s}}) \leq \epsilon. \tag{4}$$

Intuitively, the proposed constraint is related in spirit to Lipschitz continuity (Bousquet and Elisseeff, 2002), in that it bounds the sensitivity of the model's predictions to input perturbations. However, unlike classical Lipschitz continuity defined over continuous normed spaces, our constraint is expressed as an expectation over discrete, label-preserving corruptions of the input sequence. As a result, the notion of smoothness it implies is over a discrete space, rather than gradient-based smoothness over a continuous space.

The Lagrangian of the above equation can be written as follows:

$$\mathcal{L}(\theta,\lambda) = \mathcal{L}_{\text{NTP}}(\theta) + \lambda\left(\mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\tilde{\mathbf{s}}\sim\mathcal{Q}(\cdot|\mathbf{s})}\Delta_\theta(\mathbf{s},\tilde{\mathbf{s}}) - \epsilon\right) \tag{5}$$

with the constraint that the Lagrange multiplier $\lambda \geq 0$. The Lagrange multiplier controls the trade-off between fitting the training data and enforcing stability of the model's predictions under label-preserving corruptions. Different choices of λ recover familiar training objectives as special cases.

When $\lambda = 0$, the constraint is ignored and the Lagrangian reduces to

$$\mathcal{L}(\theta, 0) = \mathcal{L}_{\text{NTP}}(\theta),$$

which corresponds to standard autoregressive training. When $\lambda = 1$, ignoring the constant offset ϵ, the Lagrangian reduces to

$$\mathcal{L}_{\text{NTP}}(\theta) + \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\tilde{\mathbf{s}}\sim\mathcal{Q}}\Delta_\theta(\mathbf{s},\tilde{\mathbf{s}})$$
$$= \mathbb{E}_{\mathbf{s}\sim\mathcal{D}}\mathbb{E}_{\tilde{\mathbf{s}}\sim\mathcal{Q}}\sum_{t=1}^{T}\left(-\log p_\theta(s_t \mid \tilde{s}_{<t})\right) \tag{6}$$

This recovers the CAT objective introduced in equation (2). Hence, KItCAT corresponds to a next-token prediction objective that is constrained to learn functions that vary *smoothly* under label-preserving perturbations of the input prefix. Intuitively, adding this constraint to $\mathcal{L}_{\text{NTP}}$ should result in improved generalization.

A.4 Dataset Details

Table 2 summarizes the training data sizes and evaluation split sizes for the CPT benchmarks.

A.5 Training Details

We fine-tune all three models, *viz.*, Mistral-7B-Instruct-v0.3, Qwen3-14B, and Llama-2-7B-Chat, using Hugging Face's `TRL` library. All models are trained on four Nvidia A100 (80GB) GPUs using `bf16` precision, applying Low-Rank Adaptation (LoRA) to all linear layers. We utilize the AdamW optimizer with a warmup ratio of 0.1, training for a maximum of 64 epochs with early stopping based on validation mean token accuracy. We use LoRA rank 128 and LoRA aplha 256 throughout our experiments. For KItCAT, we treat the corruption probability p as a hyperparameter and tune it on the validation set, choosing from $\{0.10, 0.15, 0.30\}$ separately for each task. For KItCAT-SSMBA, we use ModernBERT-base as the frozen masked language model q_ϕ to sample contextually plausible replacements.

A.6 Additional Model Results

Tables 4a and 4b report the additional Qwen3-14B and Llama-2-7B-Chat experiments. All methods use the original $1\times$ corpus. These results show that the gains extend beyond a single model family and that more elaborate reconstruction or keyword-selection schemes do not consistently improve upon simple random corruption.

A.7 Compute Cost of Paraphrasing

Table 3 reports the cost in FLOPs, separating the one-time paraphrase-generation cost from training cost. The estimates use $2N$ FLOPs per generated token for a generation model with N parameters and $4M$ FLOPs per training token for LoRA training of an M-parameter model. Training steps denote the checkpoint with maximum eval mean token accuracy. Input corruption skips generation entirely and adds only a modest increase in training cost.

A.8 Sensitivity to Corruption Probability

We sweep $p \in \{0.05, 0.10, 0.15, 0.30, 0.50\}$ for both variants on Companies and Redbook 1 (Table 5). Performance has an interior optimum around $p = 0.15$–0.30 and degrades when corruption is too weak or too strong. Sensitivity is more pronounced on the data-scarce Companies corpus, where corrupting too little will result in overfitting and corrupting too much will cause confusion on what to learn.

Dataset	Train 1×	Train 10× Rephrase	Train 10× Entigraph	Dev	Test
Comp.	84.8K	396.6K	833.2K	195	689
RB1	25.4K	94.1K	247.2K	515	425
RB2	78.0K	301.2K	736.2K	1000	2269

Table 2: Dataset statistics for the CPT benchmarks. Training columns report token counts, while Dev and Test report number of examples.

Method	Steps	C_{para}	C_{train} PFLOPs	Total	×NTP	Accuracy
NTP	40	0	20	20	1.0	13.6
KItCAT-mask	60	0	30	30	1.5	27.6
KItCAT-rand	60	0	30	30	1.5	26.4
KItCAT-MASKER	50	0	25	25	1.2	17.7
KItCAT-SSMBA	150	0	74	74	3.7	21.2
Rephrase-10×	100	75	123	198	9.9	27.4
Entigraph-10×	250	180	115	295	14.8	36.1
KItCAT+Rephrase-10×	180	75	222	297	14.8	43.4
KItCAT+Entigraph-10×	250	180	115	295	14.8	**53.1**

Table 3: Estimated compute and LLM-as-a-Judge accuracy (×100) on Companies. C_{para} and C_{train} denote paraphrase-generation and training compute. Paraphrasing model: gpt-oss-120B, Base model: Mistral-7B-Instruct-v0.3,

Table 4: LLM-as-a-Judge accuracy (×100) with additional model families. Comp.: Companies; RB1/2: Redbook 1/2.

Method	Comp.	RB1	RB2	PopQA	Avg
Instruct	6.4	35.1	15.4	5.4	15.6
NTP	9.1	36.0	17.9	8.0	17.8
KItCAT-SSMBA	7.8	35.3	19.2	5.4	16.9
KItCAT-MASKER	10.4	38.4	18.4	**8.7**	19.0
KItCAT-mask	**12.9**	**39.8**	18.6	8.2	**19.9**
KItCAT-rand	11.0	37.2	**19.3**	6.3	18.5

(a) Qwen3-14B.

Method	Comp.	RB1	RB2	PopQA	Avg
Instruct	4.1	17.9	8.0	5.8	8.9
NTP	9.9	24.2	10.1	11.3	13.9
KItCAT-SSMBA	3.9	24.9	10.9	5.6	11.3
KItCAT-MASKER	11.8	**25.2**	**11.1**	**11.6**	**14.9**
KItCAT-mask	13.9	**25.2**	10.2	9.9	14.8
KItCAT-rand	**14.9**	22.6	10.9	9.3	14.4

(b) Llama-2-7B-Chat.

Dataset	Method	$p = .05$	$p = .10$	$p = .15$	$p = .30$	$p = .50$
Companies	KItCAT-rand	20.2	17.7	24.4	25.7	17.4
	KItCAT-mask	18.6	21.2	25.8	**27.1**	19.9
Redbook 1	KItCAT-rand	39.5	37.9	39.3	39.3	34.4
	KItCAT-mask	35.3	37.4	**40.9**	**40.9**	38.4

Table 5: Sensitivity to the corruption probability p on Mistral-7B. Scores are LLM-as-a-Judge accuracy (×100).

A.9 Judge Prompts

The exact prompt used for LLM-as-a-Judge evaluation with gpt-oss-120B is shown below.

LLM-as-a-Judge Evaluation Prompt

```
System:
You are an evaluator. Your task is to compare a Ground-truth Answer and a
    Prediction to decide if the Prediction correctly answers the given
    Question.

Evaluation Rules:

(1) Correctness: A correct prediction must include all essential information
    from the Ground-truth Answer. Extra information is allowed if it does not
    contradict the Ground-truth. If the Prediction states something as a
    possibility, treat it as a definitive statement.
(2) Function, Tool Names, and API Calls: If the Ground-truth Answer contains
    specific function names, tool names, API calls, or exact command
    identifiers, the Prediction must contain the same identifier(s) or clearly
     equivalent forms. Minor syntactic or formatting variations that do not
    change meaning should be treated as equivalent. For example, leading flag
    prefixes such as -, --, or no prefix at all when they clearly refer to the
     same option; underscore vs hyphen differences in identifiers when the
    intent is identical; surrounding punctuation or formatting differences
    such as backticks, quotes, parentheses, or code block notation; small
    whitespace differences or capitalization differences that do not change
    the identifier's meaning etc.
However, replacements that change the actual function/tool/API name, or
    substitute a different command that would change the behavior are
    considered incorrect. Do not penalize a prediction if it contains
    additional function / tool / API names as long as the ones present in the
    Ground-Truth are covered.
(3) URLs: If the Ground-truth Answer contains specific URLs, the Prediction
    should reference the same URL or an equivalent canonical form. Minor
    differences that do not change the target resource (for example, presence
    or absence of a trailing slash, or http vs https when both resolve to the
    same canonical resource) should be treated as equivalent. Altering the
    domain, path, or query such that the resource is different is incorrect.

Scoring Rules:

If the Prediction is correct according to the above rules, output <score>1</
    score>. If the Prediction is incomplete or incorrect, output <score>0</
    score>.

Output Format:

<explanation>
...
</explanation>

<score>
...
</score>

First provide reasoning inside <explanation> and </explanation> tags. Then
    output the score as specified above within <score> and </score> tags. Do
    not include any extra text outside these tags.

Human:
Question: {QUESTION}
Ground-truth Answer: {ANSWER}
Prediction: {ASSIST_ANSWER}
```

A.10 RephraseWeb Prompts for generating synthetic data

For RephraseWeb, we use the prompts proposed by (Ovadia et al., 2025) to generate multiple paraphrases of the training data. They use a common system prompt, along with nine different rephrase styles to induce diversity in the generated paraphrases. Both are pasted below.

System Prompt

You are an expert in text modification and paraphrasing. Your task:

- You will be given an input text (below).
- You must produce 1 distinct, long paraphrased version of that text.

Requirements:

1. **Retain Meaning & Facts:** Each paraphrase must preserve the original text's meaning, factual accuracy, and all specific details. Do not remove, alter, or add any factual information.
2. **Variety in Paraphrasing:** Each paraphrase should be substantially different from both the original and from each other in terms of vocabulary, sentence structure, and style.
3. **Maintain Length:** Each paraphrase should be approximately the same length as the original text.
4. **No Additions or Omissions:** Do not introduce external information or assumptions not present in the original text. Do not remove any facts present in the original.
5. **Preserve Specifics:** Do not change proper names, titles, dates, numbers, locations, direct quotes, or other specific references.
6. **Clarity and Tone:** Maintain the original clarity and intended tone. The result should read naturally and coherently.
7. **Output Format:** Return the paraphrased text directly as a single string. Do not include any preamble or explanation.
8. **Include Source:** If the source of the text is mentioned in the input (e.g., title, author, publication, etc.), you must include it in the paraphrased version.

Below, we include prompts for generating paraphrases in different styles to increase diversity in the synthetic data.

Reshuffle

Your sub-task is to rearrange the order of the sentences in the text provided below to provide a long paraphrased version. You must significantly change the sequence of sentences, ensuring the final text follows a clear, coherent, and logical structure. While reshuffling, you can make minor modifications to the wording to enhance the flow and coherence of the text. Your output must adhere to the previously mentioned requirements given in the system prompt.

Reword

Your sub-task is to rephrase the text provided below, focusing on the choice of words and sentence structure. Your goal is to replace as many words and phrases as possible with synonyms or alternative expressions while maintaining the original meaning, facts, and details. You must ensure that the rephrased text reads naturally and coherently. Your output must adhere to the previously mentioned requirements given in the system prompt.

Restructure

Your sub-task is to rewrite the text provided below, focusing on the structure of the sentences. You should rephrase the text by changing the sentence structures, altering the word order, and varying the length of sentences while preserving the original meaning, facts, and details. You must ensure that the rephrased text reads naturally and coherently. Your output must adhere to the previously mentioned requirements given in the system prompt.

Tense Variation

Your sub-task is to subtly adjust the verb tenses and aspects in the text provided below. Where appropriate, change some simple past tenses to past perfect, or present tenses to present continuous, while still accurately reflecting the same time frames. Do not introduce new factual content, maintain the same approximate length, and preserve the original meaning and details. Your output must adhere to the previously mentioned requirements given in the system prompt.

Simplify

Your task is to simplify the provided text by using more common language, and clearer expressions. You must ensure that the rephrased text reads naturally and coherently without losing any factual information or details. Your output must adhere to the requirements outlined in the system prompt.

Invert

Your sub-task is to produce a long paraphrased version of the provided text by inverting its structure. Specifically, you must:

- Reverse the overall order of the sentences, starting from the end of the original text and working backward toward the beginning.
- While inverting, you may make adjustments to the wording and sentence boundaries to ensure a coherent, logical flow.
- Preserve all factual information, names, dates, and other specifics without adding, removing, or distorting any facts.
- Maintain the approximate length of the original text and reflect its general tone and clarity.
- Ensure the final output reads naturally, as though the text was originally structured in this inverted order.

Your output must adhere to the requirements outlined in the system prompt.

Summarization

Your task is to produce a comprehensive and highly detailed knowledge-focused summary of the provided text.
Requirements:

1. **Thoroughness and Accuracy:** Include all factual information, key points, and essential details, ensuring nothing of importance is omitted.
2. **No Alteration of Facts:** The summary must faithfully reflect the original text without adding, removing, or distorting any information.
3. **No External Content:** Do not introduce assumptions, opinions, or details not present in the original text.
4. **Lexical Variety:** Use a diverse range of vocabulary and phrasing to make the summary more engaging, while maintaining accuracy and avoiding unnecessary repetition.
5. **Preserve Tone:** Reflect the general tone and style of the original text.

Your goal is to create an extensive summary that captures the full breadth of the source material while maintaining clarity, factual integrity, and coherent organization.

Detail Emphasis

Your sub-task is to carefully review the provided text and produce a paraphrased version that highlights all minor details and subtle nuances that might otherwise be overlooked.
Requirements:

1. **Thorough Attention to Detail:** Identify and preserve every small fact, reference, or subtle hint present in the original text, ensuring nothing is lost.
2. **No Alteration of Facts:** Accurately reflect every piece of information without distorting or omitting any detail.
3. **Clarity and Coherence:** Present the paraphrased text clearly, making sure that the small details fit naturally into the narrative and contribute to overall coherence.
4. **Fidelity to Original Tone:** Maintain the original tone, length, and style, incorporating all subtle elements in a way that feels organic and readable.

Your goal is to produce a paraphrased version that gives as much importance to minor aspects as it does to major points, ensuring full fidelity to the original text.

Middle Restructure

Your sub-task is to produce a long paraphrased version of the provided text by reorganizing its structure to begin from its midpoint.
Guidelines:

1. Identify a clear midpoint in the text and start your rewriting from that point.
2. Move forward through the latter half of the text after this midpoint, maintaining logical flow and coherence.
3. Once you reach the end of the original text, continue by incorporating the initial portion (the beginning section) at the end, so that the sequence now runs from the middle to the end, and then from the start to the middle.
4. While restructuring, you may slightly modify wording and sentence boundaries to enhance clarity, coherence, and readability.
5. Preserve all factual information and details. Do not add or remove any facts. Names, dates, locations, and other specifics must remain accurate.
6. Reflect the original tone and style.

Your output must adhere to the requirements outlined in the system prompt.

A.11 EntiGraph Generation Prompts

This appendix details the prompts proposed by (Yang et al., 2025) for generating synthetic data. The pipeline consists of three stages: (1) entity extraction from source documents, (2) two-entity relation generation, and (3) three-entity relation generation. All prompts are issued to GPT-4o via the Azure OpenAI API.

A.11.1 Entity Extraction

The following system prompt is used to extract salient entities from each source document. The model is instructed to return structured JSON.

System Prompt: Entity Extraction

```
System:
As a knowledge analyzer, your task is to dissect and understand an
article provided by the user. You are required to perform the
following steps:

1. Summarize the Article:
Provide a concise summary of the entire article, capturing the main
```

```
points and themes.

2. Extract Entities:
Identify and list all significant "nouns" or entities mentioned within
the article. These entities should include but are not limited to:

* People:
Any individuals mentioned in the article, using the names or
references provided.

* Places:
Both specific locations and abstract spaces relevant to the content.

* Objects:
Any concrete object that is referenced by the provided content.

* Concepts:
Any significant abstract ideas or themes that are central to the
article's discussion.

Try to exhaust as many entities as possible. Your response should be
structured in JSON format to organize the information effectively.
Ensure that the summary is brief yet comprehensive, and the list of
entities is detailed and accurate.

Use the following response format:

{
  "summary": "<A concise summary of the article>",
  "entities": ["entity1", "entity2", ...]
}
```

The user message for entity extraction takes the following form:

User Prompt: Entity Extraction

```
Human:
### Document Content:
{document_content}
```

A.11.2 Two-Entity Relation Generation

For each pair of extracted entities (e_i, e_j), the following system prompt instructs the model to rephrase the document content with emphasis on each entity and analyze their interaction.

System Prompt: Two-Entity Relation Generation

```
System:
You will act as a knowledge analyzer tasked with dissecting an article
provided by the user. Your role involves two main objectives:

1. Rephrasing Content:
The user will identify two specific entities mentioned in the article.
You are required to rephrase the content of the article twice:

* Once, emphasizing the first entity.
* Again, emphasizing the second entity.

2. Analyzing Interactions:
Discuss how the two specified entities interact within the context of
the article.

Your response should clearly separate the rephrased content from the
```

```
interaction analysis. Ensure each section includes sufficient context,
ideally referencing the article title to maintain clarity about the
discussion's focus.

Use the following response format:

### Discussion of <title> in relation to <entity1>
<Rephrased content focusing on the first entity>

### Discussion of <title> in relation to <entity2>
<Rephrased content focusing on the second entity>

### Discussion of Interaction between <entity1> and <entity2>
in context of <title>
<Discussion on how the two entities interact within the article>
```

The user message for two-entity relation generation takes the following form:

User Prompt: Two-Entity Relation Generation

```
Human:
### Document Content:
{document_content}

### Entities:
- {entity1}
- {entity2}
```

A.11.3 Three-Entity Relation Generation

For each triple of extracted entities (e_i, e_j, e_k), the following system prompt extends the two-entity setting to three entities.

System Prompt: Three-Entity Relation Generation

```
System:
You will act as a knowledge analyzer tasked with dissecting an article
provided by the user. Your role involves three main objectives:

1. Rephrasing Content:
The user will identify three specific entities mentioned in the
article. You are required to rephrase the content of the article three
times:

* Once, emphasizing the first entity.
* Again, emphasizing the second entity.
* Lastly, emphasizing the third entity.

2. Analyzing Interactions:
Discuss how these three specified entities interact within the context
of the article.

Your response should clearly separate the rephrased content from the
interaction analysis. Ensure each section includes sufficient context,
ideally referencing the article title to maintain clarity about the
discussion's focus.

Use the following response format:

### Discussion of <title> in relation to <entity1>
<Rephrased content focusing on the first entity>

### Discussion of <title> in relation to <entity2>
```

```
<Rephrased content focusing on the second entity>

### Discussion of <title> in relation to <entity3>
<Rephrased content focusing on the third entity>

### Discussion of Interaction between <entity1>, <entity2>, and
<entity3> in context of <title>
<Discussion on how the three entities interact within the article>
```

The user message for three-entity relation generation takes the following form:

User Prompt: Three-Entity Relation Generation

```
Human:
### Document Content:
{document_content}

### Entities:
- {entity1}
- {entity2}
- {entity3}
```