TiC-LM: A Multi-Year Benchmark for Continual Pretraining of Language Models

Jeffrey Li[2†*] **Mohammadreza Armandpour**[1*] **Iman Mirzadeh**[1] **Sachin Mehta**[1] **Vaishaal Shankar**[1]

Raviteja Vemulapalli[1] **Oncel Tuzel**[1] **Mehrdad Farajtabar**[1] **Hadi Pour Ansari**[1] **Fartash Faghri**[1∘]

[1]Apple [2]University of Washington
jwl2162@cs.washington.edu, fartash@apple.com

Abstract

Large language models (LLMs) are trained on data crawled over many years from the web. We investigate how quickly LLMs become outdated over time and how to best update them with newer data. Specifically, we simulate a world in which the latest dump of Common Crawl (CC), the most prominent public source of pre-training data, is used every month to *continually* train an LLM. We design various dynamic evaluations from the CC data, Wikipedia, and StackExchange to measure continual learning metrics such as forgetting and forward transfer. We discover that recent DataComp-LM [28] models trained on data before 2023 have already become outdated, incurring up to 45% larger noun-perplexity on 2024 Wikipedia articles compared to pre-2023 articles (Fig. 1, left). Further, we use our setup to evaluate the effectiveness of several large-scale continual learning methods and find that replaying older data is most effective for combating forgetting: for previously seen CC dumps, it can reduce the regret on held-out loss by 60% compared to other optimizer and loss-based interventions. However, some domains evolve more quickly than others, favoring different trade-offs between mixing old and new data.

1 Introduction

Large language models (LLMs) rely on massive amounts of data, a major portion of which comes from large-scale web-crawls that have been running over the past 10–20 years. Common Crawl (CC), the most well-known source of such data, has been active since 2007 and continues to release monthly dumps of data. While typically many (or all) previous dumps are combined together to train LLMs from scratch [28, 44], the vast costs and inherent knowledge cutoffs of LLMs raise natural questions about how they can be most effectively updated as future dumps are released. In this work, we introduce a benchmark for Time-Continual Learning of Language Models (TiC-LM) and investigate how to continually train LLMs over many months and years. Taking inspiration from the recent TiC-CLIP [13] work, our goal is to find efficient alternatives to training LLMs from scratch by reusing and updating prior pre-trained models. Overall, we seek to answer the following:

- How quickly do pretrained language models become outdated on new data?
- Can continual pretraining match training from scratch for the same number of tokens?
- Do forgetting and forward transfer vary across domains such as Wikipedia, News, etc?

TiC-LM centers around TiC-CommonCrawl (TIC-CC), a massive time-stratified set of training and evaluation data created using 114 CC dumps spanning 2013–2024. We also create domain-specific evaluations sourced from outside Common Crawl including TiC-Wikipedia (TIC-WIKI),

[*]Equal contribution. [∘] Project lead. [†] Work done during an internship at Apple.

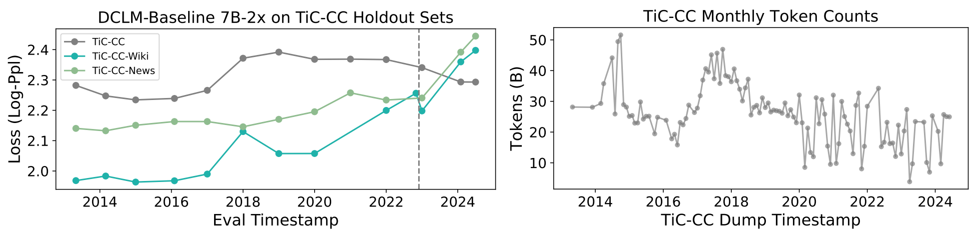


Figure 1: (Left) Performance of a model trained on DCLM-Baseline [28], which contains data up to 2022. Notably, the loss gap to our Oracle increases significantly on the Wiki and News subsets after the 2022 data cutoff (dotted line). (Right) Total number of tokens per month in TiC-CC.

and TiC-StackExchange (TiC-StackE) spanning 2008–2024. Using our benchmark, we evaluate several continual learning baselines and find that cyclic learning rate schedules and data replay can be effective for balancing learning on new data and preventing forgetting. However, we also find that different domains evolve at different rates, benefiting from more or less replay.

Related Work. Learning from multiple, sequentially observed, distributions has long been an active area of ML research [54]. More recently, several works have studied continual learning for LLMs [55], targeting improvements on: (1) general capabilities [14, 19, 43]; (2) specific domains [7, 16, 23]; (3) newer data as the world evolves [21, 20, 23, 27, 31, 33, 42, 46]. Works in this third category have shown that in several domains, the performance of LLMs decays as training and test sets grow farther apart in time, motivating the need to *efficiently* and *non-distruptively* adapt to temporal distribution shifts. Our work scales up these efforts to more closely match current LLM training practices. While older works often focus on continual training runs involving individual sources (e.g., news, Wikipedia, and social media) and <10 timesteps (see Tab. 3), we consider training on a *generic web-crawl* (i.e., Common Crawl) spanning 114 different months. In turn, the generality of TiC-CC allows us to go beyond single-domain evaluations. We provide an extended discussion of related works in Appx. E.

2 TiC-CommonCrawl: More than a Decade of Web Data

We create a large *time-stratified* dataset of 2.9T tokens based upon Common Crawl (CC), a free and open corpus of web-crawled data that releases new snapshots roughly every month. We collect all dumps between May-2013–July-2024, resulting in 114 corresponding splits that we refer to by the month of their release date. For each split, we then apply a pre-processing pipeline based on that of DataComp-LM [28]. Notably, we do not perform any operations on a particular month that depend on future months to retain causality and temporal order (see Appx. A for further details).

Data Processing. We use assets from DataComp-LM [28], starting with DCLM-Pool, which contains all CC dumps between May-2013 and Dec-2022, pre-extracted with `resiliparse` [5]. We split this data by month and reuse the same download and extraction scripts to extend DCLM-Pool to July-2024. Next, we follow DCLM-Baseline's pipeline by applying heuristic filters from RefinedWeb [44] and a fuzzy-deduplication step which we modify to run only *within* each month rather than globally. Also, we skip the final classifier-based filter in DCLM-Baseline, as this classifier was trained on data from all months. Finally, we leverage the fact that DCLM-Pool was randomly partitioned into ten equally-sized chunks to construct hold-out sets for evaluation. In Fig. 1, we show the number of tokens yielded for each month of TiC-CC. In total, the dataset spans 2.9T tokens, with individual months ranging between 10B to 50B tokens.

3 Evaluations

In this section, we will discuss our time-continual evaluations that are designed both with and independent of CC data. As our focus is on continual pretraining, we focus on perplexity evaluations without instruction-tuning (see Appx. C for exact metrics). We introduce three sets of novel evaluations: TiC-CC (which includes TiC-CC-Wiki and TiC-CC-News), TiC-Wiki, TiC-StackE.

TiC-CC evaluations. We compute the loss on three monthly subsets of our CC data:

- TiC-CC: Held-out documents coming from the full distribution for each month of TiC-CC.
- TiC-CC-Wiki: Filtered TiC-CC for English Wikipedia pages whose URLs contain either the domain `en.wikipedia.org` or `simple.wikipedia.org`.

- TIC-CC-NEWS: Pages in TIC-CC from a set of news sites based on WMT competitions [3].

TiC-Wikipedia (TIC-WIKI). We build upon TemporalWiki [20], which generates fact-based evaluations (based on perplexity of proper nouns) from four consecutive monthly snapshots of English Wikipedia/Wikidata and split between changed/unchanged knowledge. TIC-WIKI aims to capture a broader spectrum of knowledge evolution, extending the evaluation timespan to a full decade (2014–2024) and improving upon the matching of Wikipedia/Wikidata (see Appx. C.4).

TiC-StackExchange (TIC-STACKE). StackExchange has 182 communities that share knowledge by posting questions and answers. We measure perplexity on high-quality answers from selected sites by collecting answers that have been accepted by the question author (using the time an accepted answer appears to bin examples by month). The resulting evaluation contains examples from 2008–2024. We provide details of TIC-STACKE data processing in Appx. C.5.

4 Continual Learning Methods

The goal for TiC-LM methods is to match the performance of the *Oracle* which trains on all data (114 months) starting from random initialization for the full token budget. We consider methods from three categories: optimization-based, data replay, and regularization. Aside from average metrics across all timesteps, methods should balance forgetting and forward transfer metrics (defined in Sec. 5).

Optimization-based. In non-continual settings, LLMs are often trained with a cosine-decayed learning rate schedule which requires knowledge of total training steps ahead of time. In a continual setup, however, the number of total tokens grows over time. We consider the following approaches:

- *Cyclic Cosine decay* applies a cosine decay schedule within each training period with the same initial maximum value at the beginning of each round, as in TiC-CLIP and concurrent works [13, 19, 14].
- *Cosine decay + AR (autoregressive)* is similar to cyclic cosine decay except the maximum learning rate in each period is regressed from a single-cycle cosine decay [49].

Data replay. Data replay is a classical continual learning strategy to prevent forgetting, whereby in each training round, the model is fed a mixture of data from both older and the current timesteps. We consider the following replay strategies based on the best-performing strategies in TiC-CLIP [13]:

- For the current month t, we allocate a ratio α_t of the monthly token budget to the current month.
- For previous months, we redistribute the remaining tokens equally, i.e., each seeing $\frac{1-\alpha_t}{t-1}$.

In particular, when $\alpha_t = 1/t$, we see an equal number of tokens from all observed months. We also consider setting $\alpha_t = 1/2$ which always allocates half the token budget to the current month.

Regularization-based. These methods alter the training objective, generally by penalizing larger deviations from the previous month's model. Following TiC-CLIP, we try LwF [30] and EWC [26].

5 Experiments

Training details. We train 3B parameter language models using OpenLM. Each method observes a fixed number of 220B tokens, equivalent to 4x the Chinchilla optimal [17] amount. We assume that current practitioners are (a) likely to have access to more than enough data to train initial models; (b) unlikely to wait to obtain non-trivial performance. Hence, we front-load the total token budgets such that *half* is allocated to training on the first month, May-2013. Then, the remaining budget is split equally among the other 113 continual timesteps. Hyperparameter details are in Appx. B.

Evaluation metrics. Each continual run produces a $T_t \times T_e$ matrix of evaluations E where T_t, T_e are the total number of training/evaluation timesteps, E_{ij} is the performance of the model trained after training on data up to month i and evaluated on the month j. To control for inherent difficulty gaps across evaluation months, we measure the *regret* $R_{ij} = E_{i,j} - E^*_j$ where E^*_j is the performance of the *Oracle* trained on all months on month j. We subtract E^*_j instead of E_{jj} to avoid the misleadingly good forward/backward metrics if $E_{j,j}$ is bad. Following Garg et al. [13], we focus on the following summary metrics defined below assuming $T_t = T_e = T$ (deferring the $T_t \neq T_e$ case to Appx. C):

- In-distribution performance: averages along the matrix diagonal, i.e., $\sum_{i=1}^{T} = R_{ii}/T$.
- Backward transfer: averages the lower triangular of R, i.e., $\sum_{i=1}^{T} \sum_{j<i} \frac{R_{ij}}{T(T-1)/2}$.
- Forward transfer: averages the upper triangular of R analogously to backward transfer.

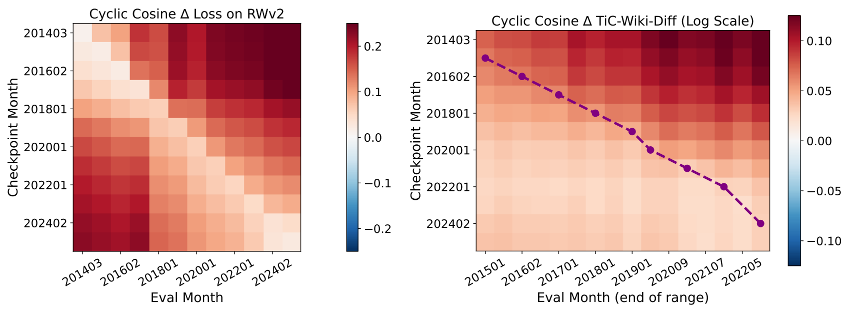


Figure 2: Eval matrices for "Cyclic Cosine" on TiC-CC and TiC-Wiki. For the latter, purple traces the ID entries. While we train on all 114 months, we subsample evaluations to be roughly annual.

Table 1: **Loss-based evaluations.** We report results relative to the Oracle. **Bold** values are within one standard deviation of the best method, estimated from three runs of Cyclic Cosine.

Method	TiC-CC ↓			TiC-CC-Wiki ↓			TiC-CC-News ↓		
	Backward	ID	Forward	Backward	ID	Forward	Backward	ID	Forward
Cyclic Cosine (std)	0.072 (0.000)	**0.027** (0.000)	**0.161** (0.000)	0.038 (0.000)	0.032 (0.000)	0.074 (0.000)	0.058 (0.000)	0.015 (0.000)	0.109 (0.000)
Cyclic Cosine + AR	0.058	0.040	0.166	0.032	0.031	0.074	0.041	0.017	0.110
Replay ($\alpha_t = 1/t$)	**0.023**	0.074	0.178	0.020	0.036	0.078	0.005	0.035	0.117
Replay ($\alpha_t = 1/2$)	0.024	0.042	0.167	0.024	0.031	0.074	0.013	0.019	0.111
Replay ($\alpha_t = 1/t$) + AR	0.026	0.083	0.181	**0.019**	0.037	0.079	**0.004**	0.039	0.119
Replay ($\alpha_t = 1/2$) + AR	0.025	0.055	0.171	0.022	0.032	0.076	0.009	0.022	0.112
LwF	0.072	**0.027**	**0.161**	0.038	0.032	0.074	0.058	0.015	0.109
EWC	0.061	0.032	0.162	0.031	**0.029**	**0.071**	0.046	**0.014**	**0.108**

Table 2: **Selected downstream evaluations.** For all dynamic evaluations, we report perplexity values relative to the Oracle with log-scaling. **Bold** values are within one standard deviation of the best.

Method	TiC-Wiki-Diff ↓			TiC-StackOverflow ↓			TiC-StackE-Math↓		
	Backward	ID	Forward	Backward	ID	Forward	Backward	ID	Forward
Cyclic Cosine (std)	0.033 (0.000)	0.052 (0.000)	0.085 (0.000)	0.041 (0.002)	0.078 (0.002)	**0.156** (0.003)	0.036 (0.001)	0.023 (0.001)	0.014 (0.001)
Cyclic Cosine + AR	0.033	0.054	0.087	**0.032**	**0.077**	0.159	0.011	0.006	0.003
Replay ($\alpha_t = 1/t$)	0.038	0.063	0.091	0.075	0.121	0.191	-0.009	-0.010	-0.006
Replay ($\alpha_t = 1/2$)	0.032	0.055	0.086	0.055	0.094	0.170	0.010	0.003	0.001
Replay ($\alpha_t = 1/t$) + AR	0.039	0.063	0.092	0.066	0.119	0.193	**-0.019**	**-0.015**	**-0.008**
Replay ($\alpha_t = 1/2$) + AR	0.033	0.057	0.088	0.047	0.096	0.176	-0.006	-0.006	-0.002
LwF	0.033	0.053	0.085	0.037	**0.075**	**0.155**	0.034	0.021	0.013
EWC	**0.030**	**0.051**	**0.083**	**0.033**	**0.077**	0.162	0.016	0.009	0.006

TiC-CC evaluations. Overall, as seen in Fig. 2 and Tab. 1, a standard cyclic cosine schedule results in significant forgetting even one year later, more prominently for TiC-CC and TiC-CC-News. Algorithmic interventions incur various trade-offs between backward transfer and ID performance. Specifically, AR meta-schedules and EWC somewhat reduce forgetting. However, as shown in the corresponding heatmaps (Appx. D), these approaches still result in significant forgetting at later checkpoints. To further reduce forgetting, using replay is required, with $\alpha_t = 1/2$ offering similar improvements to backward transfer compared to $\alpha_t = 1/t$ but with much better ID performance.

Downstream evaluations. Table 2 presents results for TiC-Wiki and TiC-StackE. On TiC-Wiki, Cyclic Cosine is best, suggesting that the impact of forgetting older CC dumps is minimal. In contrast, for TiC-StackE-Math, earlier CC dumps (before Feb-2016) appear to be most useful (Appx. D), leading to improvements from both replay and AR schedules. For TiC-StackOverflow, there not only exists a larger distribution shift over time, but seeing less old data improves all summary metrics.

6 Conclusion

We introduce a benchmark for LLM pretraining data spanning more than a decade. TiC-CC consists of training and evaluation data spanning more than 100 months over 11 years. We also introduce new TiC evaluations, TiC-Wiki, and TiC-StackE. Using these assets, we clearly observe models need to be continually trained to stay up to date but that the ideal update frequency varies per domain, motivating the need for forgetting prevention. To this end, we compared baseline strategies for continual pretraining, finding that simple cyclic learning rate schedules and data-replay shrink the gap to an *Oracle* that trains on all data. However, completely closing the gap remains an open and challenging problem to be studied by future work on our benchmark.

References

[1] Oshin Agarwal and Ani Nenkova. Temporal effects on pre-trained models for language processing tasks. *Transactions of the Association for Computational Linguistics*, 10:904–921, 2022.

[2] Yogesh Balaji, Mehrdad Farajtabar, Dong Yin, Alex Mott, and Ang Li. The effectiveness of memory replay in large scale continual learning. *arXiv preprint arXiv:2010.02418*, 2020.

[3] Loïc Barrault, Magdalena Biesialska, Ondřej Bojar, Marta R. Costa-jussà, Christian Federmann, Yvette Graham, Roman Grundkiewicz, Barry Haddow, Matthias Huck, Eric Joanis, Tom Kocmi, Philipp Koehn, Chi-kiu Lo, Nikola Ljubešić, Christof Monz, Makoto Morishita, Masaaki Nagata, Toshiaki Nakazawa, Santanu Pal, Matt Post, and Marcos Zampieri. Findings of the 2020 conference on machine translation (WMT20). In Loïc Barrault, Ondřej Bojar, Fethi Bougares, Rajen Chatterjee, Marta R. Costa-jussà, Christian Federmann, Mark Fishel, Alexander Fraser, Yvette Graham, Paco Guzman, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, André Martins, Makoto Morishita, Christof Monz, Masaaki Nagata, Toshiaki Nakazawa, and Matteo Negri, editors, *Proceedings of the Fifth Conference on Machine Translation*, pages 1–55, Online, November 2020. Association for Computational Linguistics. URL `https://aclanthology.org/2020.wmt-1.1`.

[4] Himanshu Beniwal, Mayank Singh, et al. Remember this event that year? assessing temporal information and reasoning in large language models. *arXiv preprint arXiv:2402.11997*, 2024.

[5] Janek Bevendorff, Benno Stein, Matthias Hagen, and Martin Potthast. Elastic ChatNoir: Search Engine for the ClueWeb and the Common Crawl. In Leif Azzopardi, Allan Hanbury, Gabriella Pasi, and Benjamin Piwowarski, editors, *Advances in Information Retrieval. 40th European Conference on IR Research (ECIR 2018)*, Lecture Notes in Computer Science, Berlin Heidelberg New York, March 2018. Springer.

[6] Janek Bevendorff, Martin Potthast, and Benno Stein. FastWARC: Optimizing Large-Scale Web Archive Analytics. In Andreas Wagner, Christian Guetl, Michael Granitzer, and Stefan Voigt, editors, *3rd International Symposium on Open Search Technology (OSSYM 2021)*. International Open Search Symposium, October 2021.

[7] Jie Chen, Zhipeng Chen, Jiapeng Wang, Kun Zhou, Yutao Zhu, Jinhao Jiang, Yingqian Min, Wayne Xin Zhao, Zhicheng Dou, Jiaxin Mao, Yankai Lin, Ruihua Song, Jun Xu, Xu Chen, Rui Yan, Zhewei Wei, Di Hu, Wenbing Huang, and Ji-Rong Wen. Towards effective and efficient continual pre-training of large language models. *arXiv preprint arXiv:2407.18743*, 2024. URL `https://arxiv.org/abs/2407.18743`.

[8] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=jznbgiynus`.

[9] Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273, 2022.

[10] Felix Drinkall, Eghbal Rahimikia, Janet B. Pierrehumbert, and Stefan Zohren. Time machine GPT. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Findings of the Association for Computational Linguistics: NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3281–3292. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.FINDINGS-NAACL.208. URL `https://doi.org/10.18653/v1/2024.findings-naacl.208`.

[11] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[12] Bahare Fatemi, Mehran Kazemi, Anton Tsitsulin, Karishma Malkan, Jinyeong Yim, John Palowitch, Sungyong Seo, Jonathan Halcrow, and Bryan Perozzi. Test of time: A benchmark for evaluating llms on temporal reasoning. *arXiv preprint arXiv:2406.09170*, 2024.

[13] Saurabh Garg, Mehrdad Farajtabar, Hadi Pouransari, Raviteja Vemulapalli, Sachin Mehta, Oncel Tuzel, Vaishaal Shankar, and Fartash Faghri. Tic-clip: Continual training of clip models. In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024. URL `https://openreview.net/forum?id=TLADT8Wrhn`.

[14] Kshitij Gupta, Benjamin Thérien, Adam Ibrahim, Mats Leon Richter, Quentin Gregory Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. Continual pre-training of large language models: How to re-warm your model? In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023. URL `https://openreview.net/forum?id=pg7PUJe0Tl`.

[15] Wes Gurnee and Max Tegmark. Language models represent space and time. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL `https://openreview.net/forum?id=jE8xbmvFin`.

[16] Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. Don't stop pretraining: Adapt language models to domains and tasks. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.740. URL `https://aclanthology.org/2020.acl-main.740`.

[17] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. *arXiv preprint arXiv:2302.00487*, 2022. URL `https://arxiv.org/abs/2203.15556`.

[18] Yen-Chang Hsu, Yen-Cheng Liu, Anita Ramasamy, and Zsolt Kira. Re-evaluating continual learning scenarios: A categorization and case for strong baselines. *arXiv preprint arXiv:1810.12488*, 2018.

[19] Adam Ibrahim, Benjamin Thérien, Kshitij Gupta, Mats L. Richter, Quentin Gregory Anthony, Eugene Belilovsky, Timothée Lesort, and Irina Rish. Simple and scalable strategies to continually pre-train large language models. *Trans. Mach. Learn. Res.*, 2024, 2024. URL `https://openreview.net/forum?id=DimPeeCxKO`.

[20] Joel Jang, Seonghyeon Ye, Changho Lee, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, and Minjoon Seo. Temporalwiki: A lifelong benchmark for training and evaluating ever-evolving language models. *arXiv preprint arXiv:2204.14211*, 2022.

[21] Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. Towards continual knowledge learning of language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=vfsRB5MImo9`.

[22] Zhen Jia, Philipp Christmann, and Gerhard Weikum. Tiq: A benchmark for temporal question answering with implicit time constraints. In *Companion Proceedings of the ACM on Web Conference 2024*, pages 1394–1399, 2024.

[23] Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew O. Arnold, and Xiang Ren. Lifelong pretraining: Continually adapting language models to emerging corpora. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 4764–4780. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.351. URL `https://doi.org/10.18653/v1/2022.naacl-main.351`.

[24] Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Akari Asai, Xinyan Yu, Dragomir Radev, Noah A Smith, Yejin Choi, Kentaro Inui, et al. Realtime qa: what's the answer right now? *Advances in Neural Information Processing Systems*, 36, 2024.

[25] Yujin Kim, Jaehong Yoon, Seonghyeon Ye, Sangmin Bae, Namgyu Ho, Sung Ju Hwang, and Se-Young Yun. Carpe diem: On the evaluation of world knowledge in lifelong language models. In Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 5401–5415. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.NAACL-LONG.302. URL `https://doi.org/10.18653/v1/2024.naacl-long.302`.

[26] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[27] Angeliki Lazaridou, Adhi Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomas Kocisky, Sebastian Ruder, et al. Mind the gap: Assessing temporal generalization in neural language models. *Advances in Neural Information Processing Systems*, 34:29348–29363, 2021.

[28] Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, et al. Datacomp-lm: In search of the next generation of training sets for language models. *arXiv preprint arXiv:2406.11794*, 2024.

[29] Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. Evaluating large language models for generalization and robustness via data compression. *arXiv preprint arXiv:2402.00861*, 2024.

[30] Zhizhong Li and Derek Hoiem. Learning without forgetting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(12):2935–2947, 2018. doi: 10.1109/TPAMI.2017.2773081.

[31] Adam Liška, Tomáš Kočiský, Elena Gribovskaya, Tayfun Terzi, Eren Sezener, Devang Agrawal, Cyprien de Masson d'Autume, Tim Scholtes, Manzil Zaheer, Susannah Young, Ellen Gilsenan-McMahon Sophia Austin, Phil Blunsom, and Angeliki Lazaridou. Streamingqa: A benchmark for adaptation to new knowledge over time in question answering models. *arXiv preprint arXiv:2205.11388*, 2022.

[32] Vincenzo Lomonaco, Lorenzo Pellegrini, Pau Rodriguez, Massimo Caccia, Qi She, Yu Chen, Quentin Jodelet, Ruiping Wang, Zheda Mai, David Vazquez, et al. Cvpr 2020 continual learning in computer vision competition: Approaches, results, current challenges and future directions. *Artificial Intelligence*, 303:103635, 2022.

[33] Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-collados. TimeLMs: Diachronic language models from Twitter. In Valerio Basile, Zornitsa Kozareva, and Sanja Stajner, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 251–260, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-demo.25. URL `https://aclanthology.org/2022.acl-demo.25`.

[34] Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *arXiv preprint arXiv:2308.08747*, 2023.

[35] Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A. Smith. Time waits for no one! analysis and challenges of temporal misalignment. In Marine Carpuat, Marie-Catherine de Marneffe, and Iván Vladimir Meza Ruíz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 5944–5958. Association for Computational Linguistics, 2022. doi: 10.18653/V1/2022.NAACL-MAIN.435. URL `https://doi.org/10.18653/v1/2022.naacl-main.435`.

[36] Sanket Vaibhav Mehta, Darshan Patil, Sarath Chandar, and Emma Strubell. An empirical investigation of the role of pre-training in lifelong learning. *Journal of Machine Learning Research*, 24(214):1–50, 2023.

[37] Seyed Iman Mirzadeh, Mehrdad Farajtabar, and Hassan Ghasemzadeh. Dropout as an implicit gating mechanism for continual learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pages 232–233, 2020.

[38] Seyed Iman Mirzadeh, Mehrdad Farajtabar, Razvan Pascanu, and Hassan Ghasemzadeh. Understanding the role of training regimes in continual learning. *Advances in Neural Information Processing Systems*, 33:7308–7320, 2020.

[39] Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022. URL `https://openreview.net/forum?id=0DcZxeWfOPt`.

[40] Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D Manning, and Chelsea Finn. Memory-based model editing at scale. In *International Conference on Machine Learning*, pages 15817–15831. PMLR, 2022.

[41] Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi. Is your llm outdated? benchmarking llms & alignment algorithms for time-sensitive knowledge. *arXiv preprint arXiv:2404.08700*, 2024.

[42] Kai Nylund, Suchin Gururangan, and Noah A. Smith. Time is encoded in the weights of finetuned language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *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 2571–2587. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.141. URL `https://doi.org/10.18653/v1/2024.acl-long.141`.

[43] Jupinder Parmar, Sanjev Satheesh, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Reuse, don't retrain: A recipe for continued pretraining of language models. *arXiv preprint arXiv:2407.07263*, 2024. URL `https://arxiv.org/abs/2407.07263`.

[44] Guilherme Penedo, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only. *arXiv preprint*, 2023. `https://arxiv.org/abs/2306.01116`.

[45] Ameya Prabhu, Philip HS Torr, and Puneet K Dokania. Gdumb: A simple approach that questions our progress in continual learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16*, pages 524–540. Springer, 2020.

[46] Yujia Qin, Jiajie Zhang, Yankai Lin, Zhiyuan Liu, Peng Li, Maosong Sun, and Jie Zhou. ELLE: Efficient lifelong pre-training for emerging data. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2789–2810, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.findings-acl.220. URL `https://aclanthology.org/2022.findings-acl.220`.

[47] Anastasia Razdaibiedina, Yuning Mao, Rui Hou, Madian Khabsa, Mike Lewis, and Amjad Almahairi. Progressive prompts: Continual learning for language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/forum?id=UJTgQBc91_`.

[48] Guy D Rosin, Ido Guy, and Kira Radinsky. Time masking for temporal language models. In *Proceedings of the fifteenth ACM international conference on Web search and data mining*, pages 833–841, 2022.

[49] Karsten Roth, Vishaal Udandarao, Sebastian Dziadzio, Ameya Prabhu, Mehdi Cherti, Oriol Vinyals, Olivier Hénaff, Samuel Albanie, Matthias Bethge, and Zeynep Akata. A practitioner's guide to continual multimodal pretraining. *arXiv preprint arXiv:2408.14471*, 2024.

[50] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[51] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International conference on machine learning*, pages 4528–4537. PMLR, 2018.

[52] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[53] Tu Vu, Mohit Iyyer, Xuezhi Wang, Noah Constant, Jerry W. Wei, Jason Wei, Chris Tar, Yun-Hsuan Sung, Denny Zhou, Quoc V. Le, and Thang Luong. Freshllms: Refreshing large language models with search engine augmentation. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 13697–13720. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.findings-acl.813`.

[54] Liyuan Wang, Xingxing Zhang, Hang Su, and Jun Zhu. A comprehensive survey of continual learning: Theory, method and application, 2024. URL `https://arxiv.org/abs/2302.00487`.

[55] Tongtong Wu, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. Continual learning for large language models: A survey, 2024. URL `https://arxiv.org/abs/2402.01364`.

[56] Klim Zaporojets, Lucie-Aimée Kaffee, Johannes Deleu, Thomas Demeester, Chris Develder, and Isabelle Augenstein. Tempel: Linking dynamically evolving and newly emerging entities. *Advances in Neural Information Processing Systems*, 35:1850–1866, 2022.

[57] Bowen Zhao, Zander Brumbaugh, Yizhong Wang, Hannaneh Hajishirzi, and Noah A. Smith. Set the clock: Temporal alignment of pretrained language models. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *Findings of the Association for Computational Linguistics, ACL 2024, Bangkok, Thailand and virtual meeting, August 11-16, 2024*, pages 15015–15040. Association for Computational Linguistics, 2024. URL `https://aclanthology.org/2024.findings-acl.892`.

[58] Jonathan Zheng, Alan Ritter, and Wei Xu. NEO-BENCH: evaluating robustness of large language models with neologisms. In Lun-Wei Ku, Andre Martins, and Vivek Srikumar, editors, *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 13885–13906. Association for Computational Linguistics, 2024. doi: 10.18653/V1/2024.ACL-LONG.749. URL `https://doi.org/10.18653/v1/2024.acl-long.749`.

[59] Da-Wei Zhou, Fu-Yun Wang, Han-Jia Ye, and De-Chuan Zhan. Pycil: A python toolbox for class-incremental learning, 2023.

A Constructing TiC-CC

We build upon the existing pipeline and assets from DataComp-LM [28] to build our dataset, only altering steps that rely on global operations across months. In total, using our processing pipeline can yield up to 29T tokens. In our experiments, we produce a smaller subset of 2.9T with our training runs each using 220B. Future work can expand to training at larger scales such as using the full 2.9T or 29T tokens.

Initial pool and temporal splitting. We start with DCLM-Pool [28] which contains all CC dumps between May-2013 and December-2022. The only pre-processing that has been done on this pool is to parse the HTML (contained in WARC files of CC) into plaintext for each webpage via the open-source `resiliparse` library [5, 6]. In DCLM-Pool, documents are naturally grouped together into files based upon the CC dump, which is indicated by the file prefix [2]. To split the data by month, we simply group files that share the same prefix. Since DCLM-Pool contains data up to December-2022, we also follow their exact download and extraction scripts to obtain more recent data until July-2024.

Data preprocessing and tokenization. Next, we follow DCLM-Baseline's filtering procedure which starts with their implementation of heuristic filters from RefinedWeb. We apply these filters independently on each page with no change. However, we have to modify their deduplication that removes nearly identical paragraphs/pages given similarity thresholds. Instead of applying deduplication globally as in DCLM-Baseline, we apply the same deduplication method *only within* each month. Finally, we also skip the final classifier-based filtering in DCLM-Baseline, as their classifier was trained on data that comes from all months, including examples generated by recent LLMs such as GPT-4.

Data sampling and held-out sets. DCLM-Pool was partitioned randomly into 10 equally sized "global shards". For our training scales, using just one of these global shards within each month is sufficient. Notably though, when we construct evaluation sets such as for our holdout TiC-CC evaluations, we make sure to sample from a different global shard than the one used for training. This ensures the evaluation data is a sampled from the same distribution as the training data while also *mostly* held out. Notably, because we do not deduplicate across globals shards or months, there could be overlap between training and eval sets across months. For each validation set, we cap the maximum number of tokens to 16.7M which corresponds to 8192 sequences for our context length of 2048. For some months of TiC-CC-Wiki and TiC-CC-News, we end up with less than this amount, but the smallest are 5M and 12M respectively. Additionally, we plot in Fig. 3 (right) the percentage of TiC-CC tokens that come from TiC-CC-Wiki and TiC-CC-News (before the 16.7M token cap is applied). Interestingly, both English Wikipedia and news sites make up a much smaller fraction of later dumps.

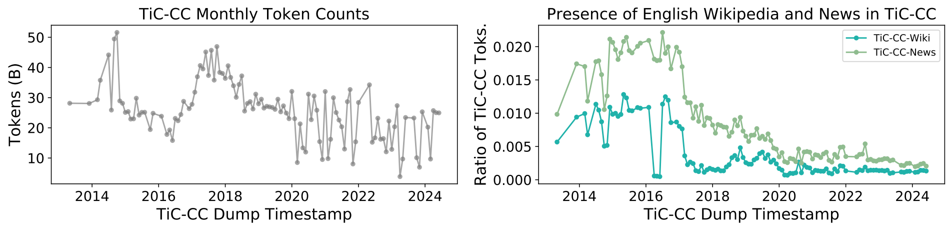


Figure 3: We plot the total number of tokens per month in TiC-CC (left) as well as the proportion of those tokens coming from our TiC-CC-Wiki and TiC-CC-News subsets (right).

[2]In DCLM-Pool, each file always starts with `CC-MAIN-YYYYMM` where `YYYYMM` indicates the dump month.

B Hyperparameter tuning

In general we follow the configurations used in DataComp-LM [28] unless further specified. For our Oracle and initialization trained on May-2013, we exactly follow their hyperparameters given that these were also standard pre-training runs from scratch.

For our various continual methods, we do perform additional hyperparameter tuning using the first 10 TiC-CC training and held-out validation set. We limit the tuning to an early set of months given that it would be impossible for a practitioner to be able to tune based upon data they have not seen far in the future. We discuss the tuning for specific methods in more detail below.

Cyclic Cosine. We mainly tuned the maximum learning rate in each cycle, trying values between 1e-3 and 5e-5. Overall we find that 1e-4 worked best, which held up even when we ran all of these configurations to completion. For the runs involving AR meta-schedules, we find that the initial maximum learning rate mattered less given that it gets decayed in later rounds.

LwF. Following the original paper [30], we used a temperature parameter of $T = 2$. We mainly tuned the regularization weight λ trying values between 0.1 and 1.0 and settling upon 0.3.

EWC. We fixed the number of iterations used to estimate the Fisher matrix to 100 and similar to LwF, we focused on tuning the weight given to the EWC regularization term. Overall, we found that fairly high values were needed to overcome the small values in the approximate Fisher matrix (coming from small second order moment terms). We found that $\lambda = 10^7$ performed best when tuning between 10^1 and 10^9.

C Details of Evaluations

C.1 Comparison with existing benchmarks

Table 3 below summarizes our proposed datasets compared with the most related time-continual benchmarks. With 2.9T tokens and evaluations across several domains, TIC-CC is the *largest and most diverse* continual learning benchmark for language model pretraining.

Table 3: Comparison with continual learning benchmarks for LLMs. CLS: classification, SUM: Summarization KB: Knowledge Base, QA: Question-Answering, LM: Language Modeling. Acc.: Accuracy, Ppl.: Perplexity, Tok.: Tokens, Art.: Articles.

Benchmark	**Domain**	**Task**	**Metric**	**CL Train**	**Time-CL**	**Years**	**Timesteps**	**# CL Train**	**# Eval Samples**
Gururangan et al. [16]	Science,News,Reviews	CLS	micro/macro-F1	✓	✗-Task CL	—	—	0.3M	140k
Luu et al. [35]	Tweet,Science,News,Reviews	CLS/SUM	F1/Rouge-L	✓	✓	2013–2022	4–7	695k	695k
Chrono. Tweet[2022]	Science,Tweet	CLS	micro/macro-F1	✓	✓	2014–2020	4	25M	4M
TempEL [2022]	Wikipedia	KB	EL Acc.	✓	✓	2013–2022	10	—	92k
TemporalWiki [2022]	Wikipedia	KB	Noun Ppl.	✗	✓	2021	4	23B Tok.	36k
StreamingQA [2022]	News	QA	Acc.	✓	✓	2007–2020	12	99k Art.	46k
EvolvingQA [2024]	Wikipedia	QA	EM/F1	✓	✓	2007–2020	6	—	46k
TIQ [2024]	Wikipedia	QA	Precision/Rank	✓	✓	1801–2025	—	6k QA	4k
TAQA [2024]	Wikipedia	QA	F1	✓	✓	2000–2023	—	9k QA	11k
TIC-CC (All/Wiki/News)	Generic Web	LM	Ppl.	✓	✓	2013–2024	114	2.9T Tok.	2.7M
TIC-WIKI	Wikipedia	KB	Noun Ppl.	✗	✓	2014–2024	62	—	10M
TIC-STACKE	Code,Math,English,...	KB / QA	Ppl.	✗	✓	2008–2024	187	—	3.65M

C.2 Perplexity based metrics

We employ three distinct perplexity metrics for different evaluations:

$$\text{ppl}_{\text{token}} = \exp\left(\frac{\sum_{d\in\mathcal{D}}\sum_{t\in T_d} -\log P(t|c_{<t})}{\sum_{d\in\mathcal{D}}|T_d|}\right), \tag{1}$$

where $\mathcal{D}$ is a set of documents, T_d is the set of tokens in document d, and $c_{<t}$ is the context prior to token t.

$$\text{ppl}_{\text{answer}} = \frac{1}{|\mathcal{Q}|}\sum_{q\in\mathcal{Q}} \exp\left(-\log P(a_q|c_q)\right), \tag{2}$$

where $\mathcal{Q}$ is a set of question-answer pairs, a_q is the gold answer for question q, and c_q is the context.

$$\text{ppl}_{\text{noun}} = \exp\left(\frac{\sum_{d\in\mathcal{D}}\sum_{n\in N_d} -\log P(n|c_{<n})}{\sum_{d\in\mathcal{D}}|N_d|}\right), \tag{3}$$

where $\mathcal{D}$ is the set of documents in a snapshot, N_d is the set of proper noun tokens (tagged as NNP or NNPS by a POS tagger) in document d, and $c_{<n}$ is the context prior to noun n.

Our proposed evaluations then map to these metrics as follows:

- TIC-CC uses $\text{ppl}_{\text{token}}$
- TIC-WIKI uses ppl_{noun}
- TIC-STACKE uses $\text{ppl}_{\text{answer}}$

where we report results after further log-scaling (unless otherwise indicated).

C.3 Summary metrics for misaligned training and and evaluation periods

As discussed in Sec. 5, when there are an equal number of training and evaluation timesteps ($T_t = T_e$) that are also aligned (as in the case of TIC-CC), our three summary metrics are simple to define.

- In-distribution (ID) performance: averages along the matrix diagonal, i.e., $\sum_{i=1}^{T} = R_{i,i}/T$.
- Backward transfer: averages the lower triangular of R, i.e., $\sum_{i=1}^{T} \sum_{j<i} \frac{R_{i,j}}{T(T-1)/2}$.
- Forward transfer: averages the upper triangular of R analogously to backward transfer.

For some downstream evaluations, the train/evaluation periods do not exactly align ($T_t \neq T_e$), making the definition of ID more nuanced. For such evaluations, we define a_i as the index of the nearest evaluation timestep that comes before the training timestep i. We then count R_{i,a_i} towards the ID average only if no other training timestep is closer to a_i (i.e., $a_i \neq a_{i-1}$). Otherwise, we count $R_{i,j}$ towards backward and forward transfer when $j < a_i$ and $j \geq a_i$ respectively.

C.4 TIC-WIKI

We construct TIC-WIKI from Wikipedia and Wikidata which are sister projects from the non-profit Wikimedia Foundation. Wikidata is a structured knowledge graph that stores the structured data of Wikipedia and other sister projects. Data on Wikidata is represented in the form of statements in the form of property-value about an item in the simplest form. For example, "Mount Everest is the highest mountain in the world" is represented as Earth (Q2) (item) $\rightarrow$ highest point (P610) (property) $\rightarrow$ Mount Everest (Q513) (value) [3]. The triplet (item, property, value) can also be referred to as (subject, relation, object).

TemporalWiki dataset generation. TemporalWiki constructs evaluations from monthly snapshots of English Wikipedia and Wikidata through the following steps:

1. Generate TWiki-Diffsets by identifying changes and additions between consecutive snapshots of Wikipedia. For new articles, the entire article is added to the Diffset while for existing articles, only the changed or new paragraphs are added.
2. Construct TWiki-Probes by processing two consecutive snapshots of Wikidata. Statements are categorized into changed if the property/value has changed or categorized into unchanged otherwise.
3. Align TWiki-Diffsets with Wikidata by ensuring changed statements exist in TWiki-Diffsets and unchanged statements exist in Wikipedia.
4. Heuristic filtering by removing statements where the subject or object is a substring of the other or the object is more than 5 words. Moreover, a single subject is limited to maximum 1% and relation/object is limited to maximum 5% of the total statements.

TIC-WIKI extends TemporalWiki in various ways:

1. Expanding the timespan from four months to a decade (2014-2024), thus capturing a broader spectrum of knowledge evolution.
2. We improve the matching process of Wikipedia and Wikidata dumps, and enhance the robustness of data parser to format changes over time.

[3] `https://www.wikidata.org/wiki/Help:About_data`

C.4.1 Data preprocessing

Wikidata and Wikipedia dumps. Wikimedia releases regular dumps [4,5], but only retains data for the most recent 4 months. To access historical data, we utilized the Internet Archive [6]. The earliest available dump dates back to November 2014. It is important to note that the archived dumps do not cover every month, with several months missing from the record. In our study, we made use of all available monthly dumps. The filenames of the dumps include the specific date of month that has been collected on, which is typically the 1st or 20th of the month, though this can vary. We include only one dump per month if multiple dumps are available. We check for the first date if not available look for 20th and if neither we start from begining the monthh and check for the first availble date in that month.

Data cleanup. We utilize WikiExtractor [7] to clean up the Wikipedia data. This step extracts the main content and removes extraneous and non-essential characters.

Wikipedia diffsets. To construct consecutive diffs of Wikipedia, we developed a method comparing snapshots of articles from consecutive dumps. For comparing two snapshots of an article, we first remove extraneous whitespace and standardize formatting by preprocessing the text. This involves removing empty lines, stripping newline characters, and creating a normalized version of each line where punctuation is removed and text is converted to lowercase.

Afterward, we use a two-level comparison: first at the paragraph level, then at the sentence level for changed paragraphs. We utilize Python's `difflib.SequenceMatcher` to compare the normalized versions of paragraphs and sentences. This hierarchical method, coupled with normalization, captures substantial edits while filtering out minor or stylistic changes.

We extract and store both changed and unchanged content separately. Changed content includes replaced paragraphs with modified sentences and newly inserted paragraphs. Unchanged content preserves paragraphs and sentences that remain identical between versions. New articles are treated as entirely changed content. This approach allows us to focus on meaningful content changes while maintaining the context of unchanged information, providing a comprehensive view of how Wikipedia articles evolve over time. Algorithms 1 and 2 describe the process of constructing Wikipedia diffs and changed/unchanged content.

Wikidata diffsets. Next, we extract changed and unchanged Wikidata statements of the form (subject, relation, object) from each consecutive dump. Identical triplets in both dumps are marked as unchanged. Triplets in the new dump not present in the old are categorized as new, with the exception that if a subject entity has more than 10 triplets, the algorithm randomly samples 10 to represent it. When a triplet has the same subject and relation as one in the old dump but a different object and the old and new objects differ only in case (upper/lowercase), the triplet is classified as unchanged; otherwise, it is categorized as new. Triplets from the old dump not found in the new one are implicitly considered removed, but importantly, these are not included in the output sets of changed or unchanged triplets. Throughout this process, the algorithm filters out triplets with overly long object values (more than 5 words) and ensures no duplicates are added. This approach efficiently tracks Wikidata evolution, capturing nuanced changes while managing the volume of data for new entities. Algorithm 3 describes the process of triplet extraction.

Wikipedia historical dumps. It is possible to reconstruct each version of Wikiepdia using the large history files Wikipeida provide [8]. There are more than 200 historical dumps of English Wikipedia, each sized more than 2GB. Combined together, these files include all revisions and all pages of Wikipeida.

For Wikidata, Wikimedia does not provide historical diff files as Wikipedia except for the last four months [9]. Wikidata file names are formatted similar to

[4] `https://dumps.wikimedia.org/wikidatawiki/`

[5] `https://dumps.wikimedia.org/enwiki/`

[6] `https://archive.org`

[7] `https://github.com/attardi/wikiextractor`

[8] `https://dumps.wikimedia.org/enwiki/latest/` file names containing `pages-meta-history`.

[9] `https://dumps.wikimedia.org/wikidatawiki/`

`wikidatawiki-20190101-pages-articles.xml.bz2` and available at URLs similar to `https://dumps.wikimedia.org/wikidatawiki/20240401/`.

Each Wikidata dump is approximately 140GB whereas each Wikipeida dump is less than 22GB. Therefore, it is possible to make a version of Wikipedia that keeps track of all changes which results in 200 files of 2GB. But as far as we know there are no such files for Wikidata.

Using the dumps from `archive.org` has several advantages:

- We are sure that we do not leak information from previous timesteps.
- There exists a Wikidata dump close to each Wikipedia dump to be aligned.
- We can use Wiki-Extractor for filtering and remove Wikipeida editorial discussions.

To illustrate the characteristics of our generated dataset, we present key statistics in the following figures. Figure 4 shows the number of Wikipedia pages with significant changes between consecutive database dumps over time. This graph provides insight into the volume and temporal distribution of our data generation process, highlighting periods of higher and lower content modification as well as distribution of our dumps.

Algorithm 1 Construct Wikipedia Consecutive Diffs

```
 1: Input: oldSnapshot, newSnapshot
 2: Output: changedContent, unchangedContent
 3: oldArticles ← ReadArticles(oldSnapshot)
 4: newArticles ← ReadArticles(newSnapshot)
 5: changedContent ← ∅, unchangedContent← ∅
 6: for each articleId in newArticles.keys do
 7:     if articleId in oldArticles then
 8:         oldText ← NormalizeText(oldArticles[articleId].text)
 9:         newText ← NormalizeText(newArticles[articleId].text)
10:         changed ← ExtractChangedContent(oldText, newText)
11:         unchanged ← ExtractUnchangedContent(oldText, newText)
12:         Add (articleId, changed) to changedContent
13:         Add (articleId, unchanged) to unchangedContent
14:     else
15:         Add (articleId, newArticles[articleId].text) to changedContent
16: return changedContent, unchangedContent
```

Algorithm 2 Extract Changed Content

```
 1: Input: $oldText$, $newText$
 2: Output: $changedContent$
 3: oldParagraphs ← SplitIntoParagraphs(oldText)
 4: newParagraphs ← SplitIntoParagraphs(newText)
 5: changedContent ← ∅
 6: for each (oldPara, newPara) in Zip(oldParagraphs, newParagraphs) do
 7:     if IsDifferent(oldPara, newPara) then
 8:         oldSentences ← SplitIntoSentences(oldPara)
 9:         newSentences ← SplitIntoSentences(newPara)
10:         for each (oldSent, newSent) in Zip(oldSentences, newSentences) do
11:             if IsDifferent(oldSent, newSent) then
12:                 Add newSent to changedContent
13: return changedContent
```

C.5 TIC-STACKE

C.5.1 Data preprocessing

TIC-STACKE spans data from July 2008 through April 2024. The data was sourced from `archive.org` using the April 2024 dump of StackExchangeEach category in the dump comes with two key

Algorithm 3 Wikidata Triplet Extraction and Categorization

```
Require: oldDump, newDump
Ensure: unchanged, new
  unchanged ← {}
  new ← {}
  newEntities ← {}
  for all triplet ∈ newDump do
    if triplet ∈ oldDump then
      Add triplet to unchanged
    else if hasSameSubjectPredicate(triplet, oldDump) then
      oldObject ← getObject(triplet.subject, triplet.predicate, oldDump)
      if equalsIgnoreCase(triplet.object, oldObject) then
        Add triplet to unchanged
      else
        Add triplet to new
    else
      if triplet.subject ∉ oldDump then
        Add triplet to newEntities[triplet.subject]
      else
        Add triplet to new
  sampleNewEntityTriplets(newEntities, new)
  filterLongObjects(unchanged, new)
  removeDuplicates(unchanged, new)
  return unchanged, new
```

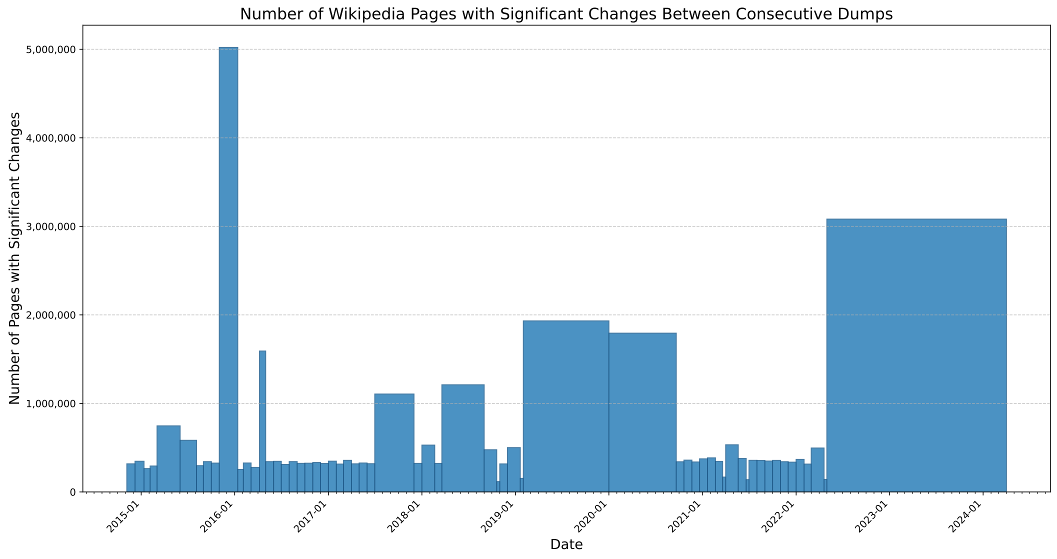


Figure 4: Number of Wikipedia pages with significant Changes between consecutive `archive.org` dumps.

files: `Post.xml` and `PostHistory.xml`. `Post.xml` contains information on how answers and questions relate to each other and includes the latest text for each post entry. `PostHistory.xml` records the changes to each post, whether it is a question or an answer.

To construct our dataset, we first build the graph of question-answer relationships based on the `Post.xml`. We then use `PostHistory.xml` to reconstruct exact snapshots of posts at specific timestamps. This allowed us to capture the state of each post at the end of each month, ensuring our data reflected the actual content available at those points in time.

We construct binary classification tasks from StackExchange content. For each question, we extract two responses: the solution accepted by the original author and an alternative option. Our goal is to create clear distinctions in answer quality, so we implement rigorous selection criteria. Specifically, we requir the accepted solution to have received at least four times the number of upvotes as the alternative. For the alternative, we choose the response with the lowest upvote count that was posted before the accepted answer. This strict filtering, while effective in creating distinct quality differentials, significantly reduced our sample size across most categories. To maintain robust evaluation metrics while preserving data volume, we introduce an additional metric: the average perplexity of accepted answers, calculated without applying the strict upvote ratio filter. This approach allows us to include more samples in our analysis while still capturing meaningful performance trends

We applied this process consistently across all categories of StackExchange, allowing for comprehensive evaluation. In total, we processed 174 out of 182 categories in stackexchange data, of which we focus on `stackoverflow` in this work as well as a group of seven categories: apple, codereview, electronics, english, gaming, math, and worldbuilding. Some categories had insufficient questions in a single month to provide statistically significant results. In such cases, we combined data from consecutive months, ensuring that each time frame contains at least 500 questions.

The full set of sites includes:

3dprinting, academia, ai, android, anime, apple, arduino, astronomy, aviation, avp, beer, bicycles, bioacoustics, bioinformatics, biology, bitcoin, blender, boardgames, bricks, buddhism, cardano, chemistry, chess, chinese, christianity, civicrm, codegolf, codereview, coffee, cogsci, computergraphics, conlang, cooking, craftcms, crafts, crypto, cs, cseducators, cstheory, datascience, dba, devops, diy, drones, drupal, dsp, earthscience, ebooks, economics, electronics, elementaryos, ell, emacs, engineering, english, eosio, esperanto, ethereum, expatriates, expressionengine, fitness, freelancing, french, gamedev, gaming, gardening, genai, genealogy, german, gis, graphicdesign, ham, hardwarerecs, health, hermeneutics, hinduism, history, homebrew, hsm, interpersonal, iot, iota, islam, italian, japanese, joomla, judaism, korean, langdev, languagelearning, latin, law, lifehacks, linguistics, literature, magento, martialarts, materials, math, matheducators, mathematica, mechanics, meta, moderators, monero, money, movies, music, musicfans, mythology, networkengineering, opendata, opensource, or, outdoors, parenting, patents, pets, philosophy, photo, physics, pm, poker, politics, portuguese, proofassistants, puzzling, quant, quantumcomputing, raspberrypi, retrocomputing, reverseengineering, robotics, rpg, rus, russian, salesforce, scicomp, scifi, security, sharepoint, sitecore, skeptics, softwareengineering, softwarerecs, solana, sound, space, spanish, sports, sqa, stackoverflow, stats, stellar, substrate, sustainability, tex, tezos, tor, travel, tridion, ukrainian, unix, ux, vegetarianism, vi, webapps, webmasters, windowsphone, woodworking, wordpress, workplace, worldbuilding, and writers.

C.5.2 Analysis of StackExchange Data

This section presents an analysis of question-answer patterns across the top 20 categories of StackExchange, with a focus on StackOverflow, Mathematics, and English Language & Usage.

Overall category distribution. Figure 5 shows the distribution of questions across the top 20 StackExchange categories.

Temporal trends in question volume. Figure 6 show the number of questions asked per month for Stack Overflow, Mathematics, and English Language & Usage.

Question characteristics. Figure 7 illustrates the distribution of question lengths for StackOverflow, Mathematics, and English Language & Usage.

Answer patterns. Figure 8 presents the distribution of answer counts per question for StackOverflow, Mathematics, and English Language & Usage.

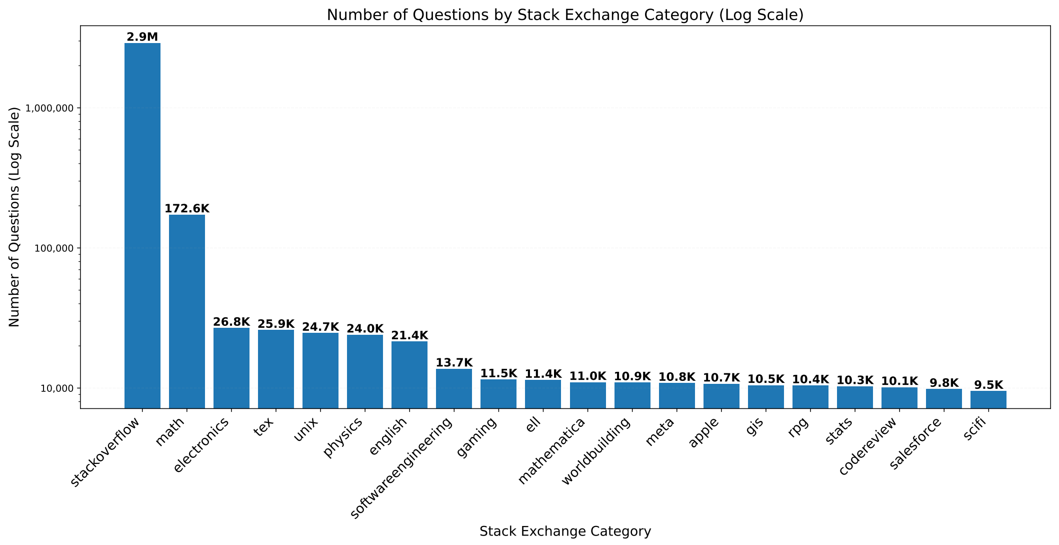
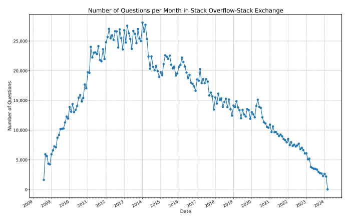
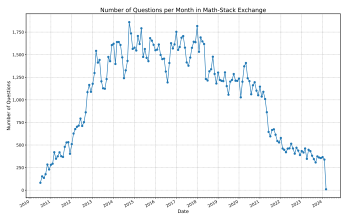


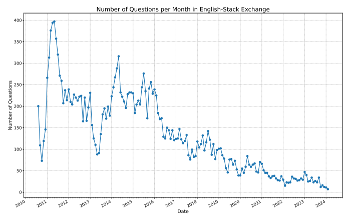
Figure 5: Number of questions by StackExchange category (log scale).



(a) StackOverflow

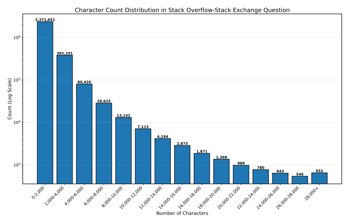


(b) Mathematics

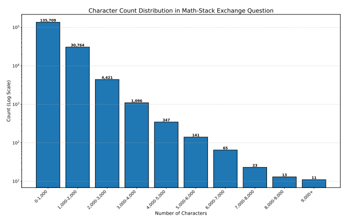


(c) English Language & Usage

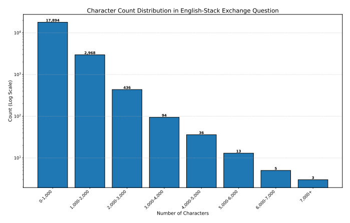
Figure 6: Number of questions per month in StackOverflow, Mathematics and English Language & Usage.



(a) StackOverflow

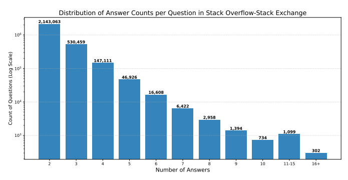


(b) Mathematics

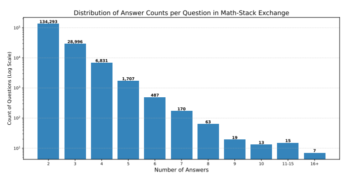


(c) English Language & Usage

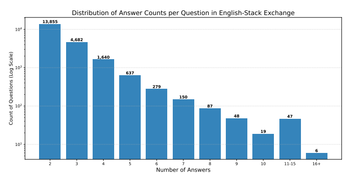
Figure 7: Character Count Distribution in StackOverflow, Mathematics and English Language & Usage Questions.



(a) StackOverflow



(b) Mathematics



(c) English Language & Usage

Figure 8: Distribution of Answer Counts per Question in Mathematics and English Language & Usage.

D Extended Results

D.1 TiC-CommonCrawl (TIC-CC) Validation sets

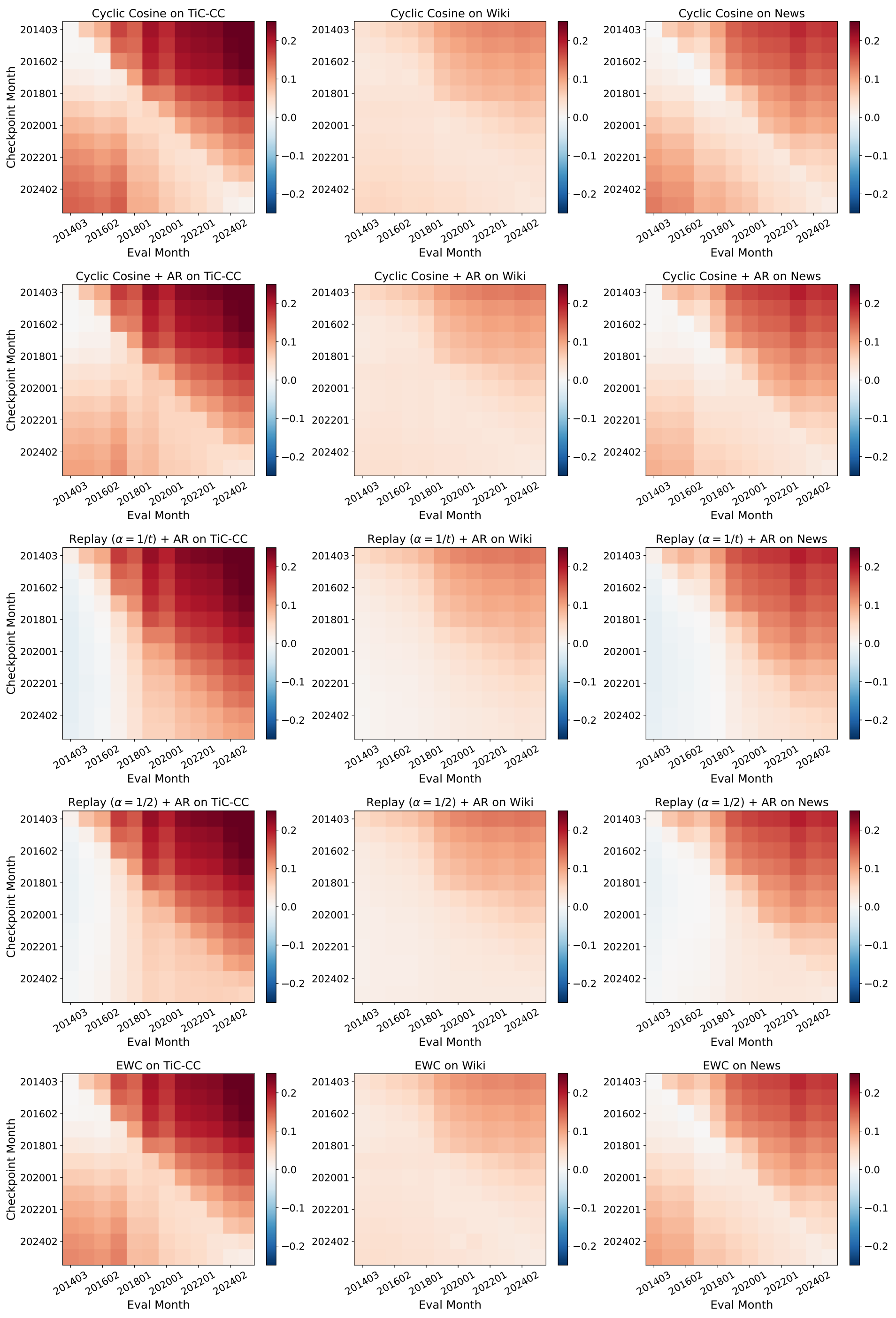


Figure 9: Evaluation matrix heatmaps for selected methods on our TIC-CC evaluations.

D.2 TiC-Wikipedia (TiC-Wiki)

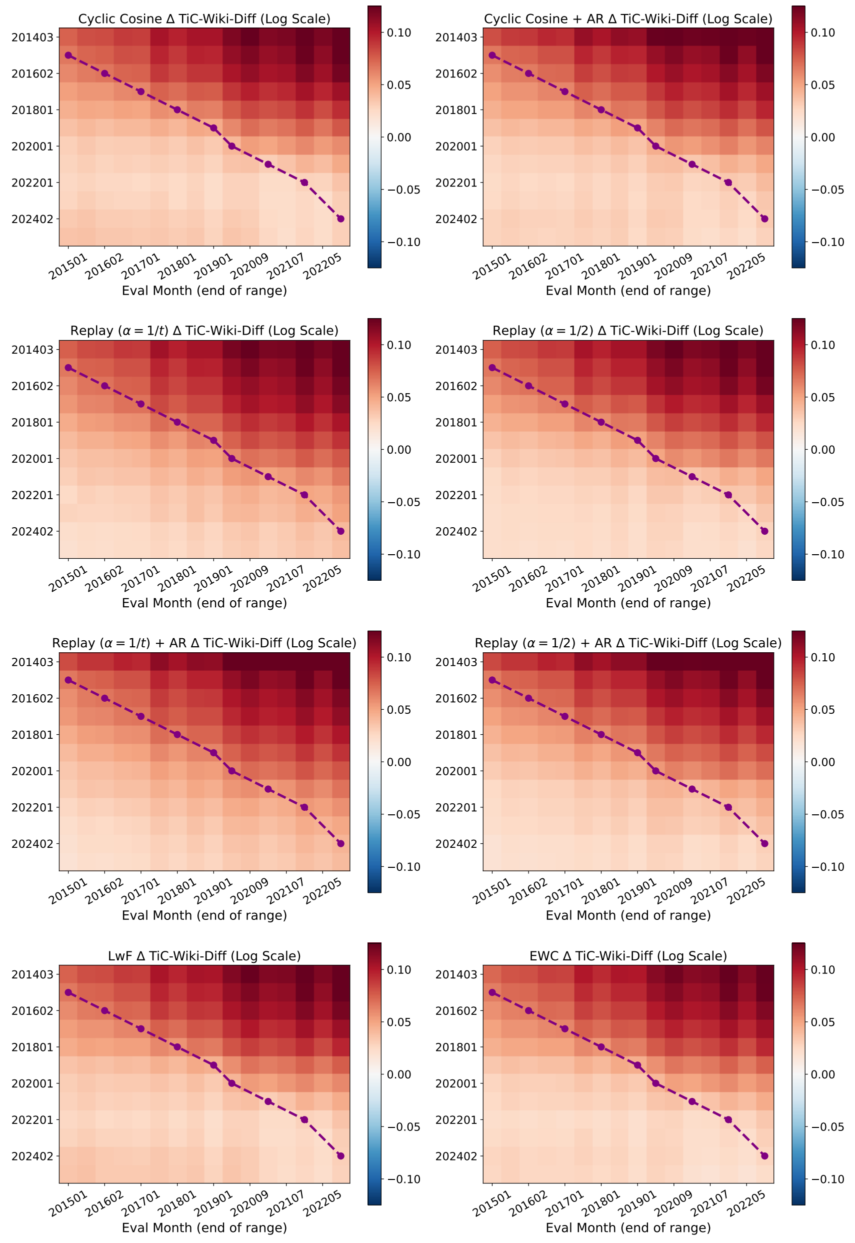


Figure 10: Evaluation matrix heatmaps for various methods on TiC-Wiki.

D.3 TiC-Stackexchange (TIC-STACKE)

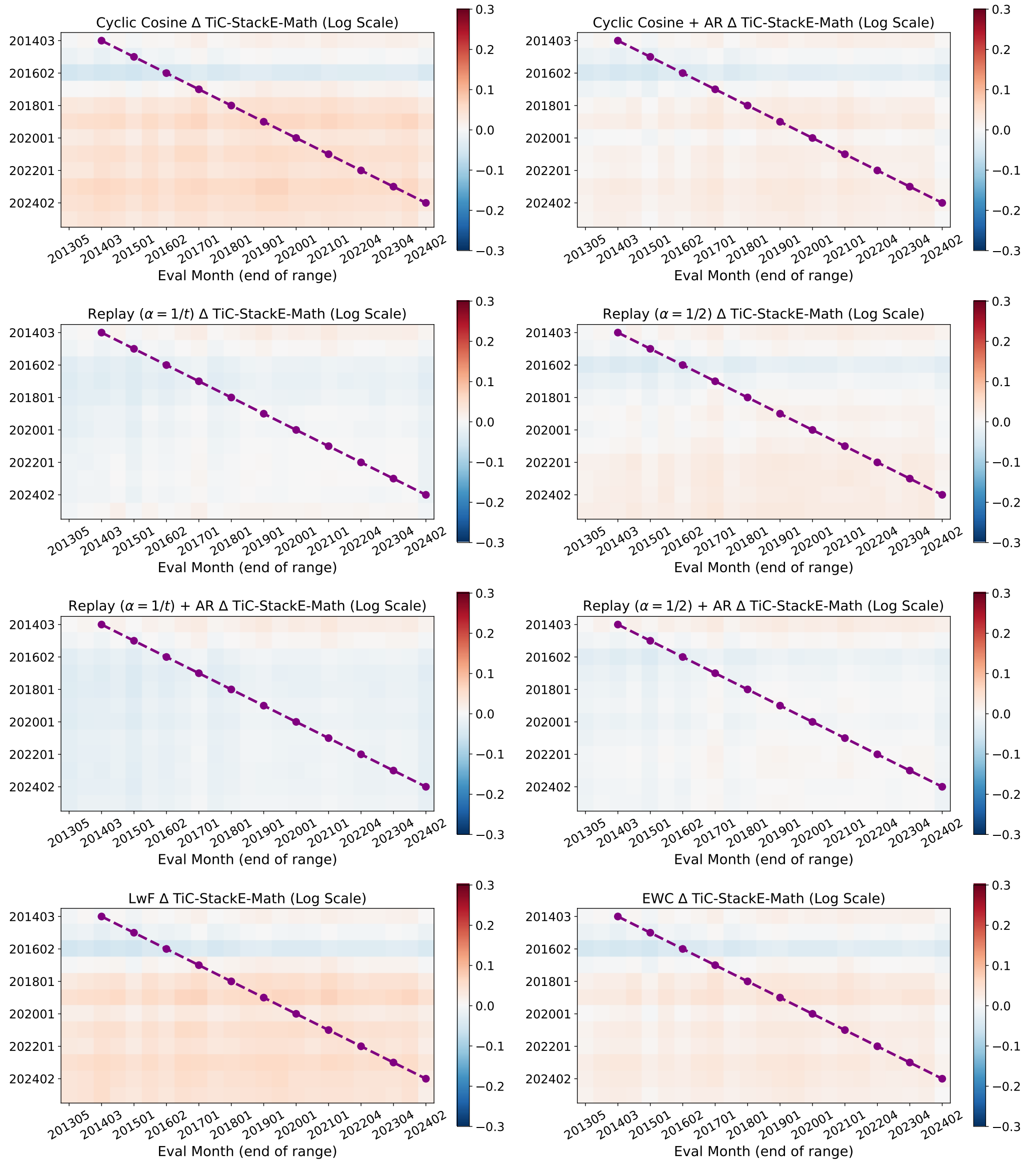


Figure 11: Evaluation matrix heatmaps for various methods on the Math site of TIC-STACKE.

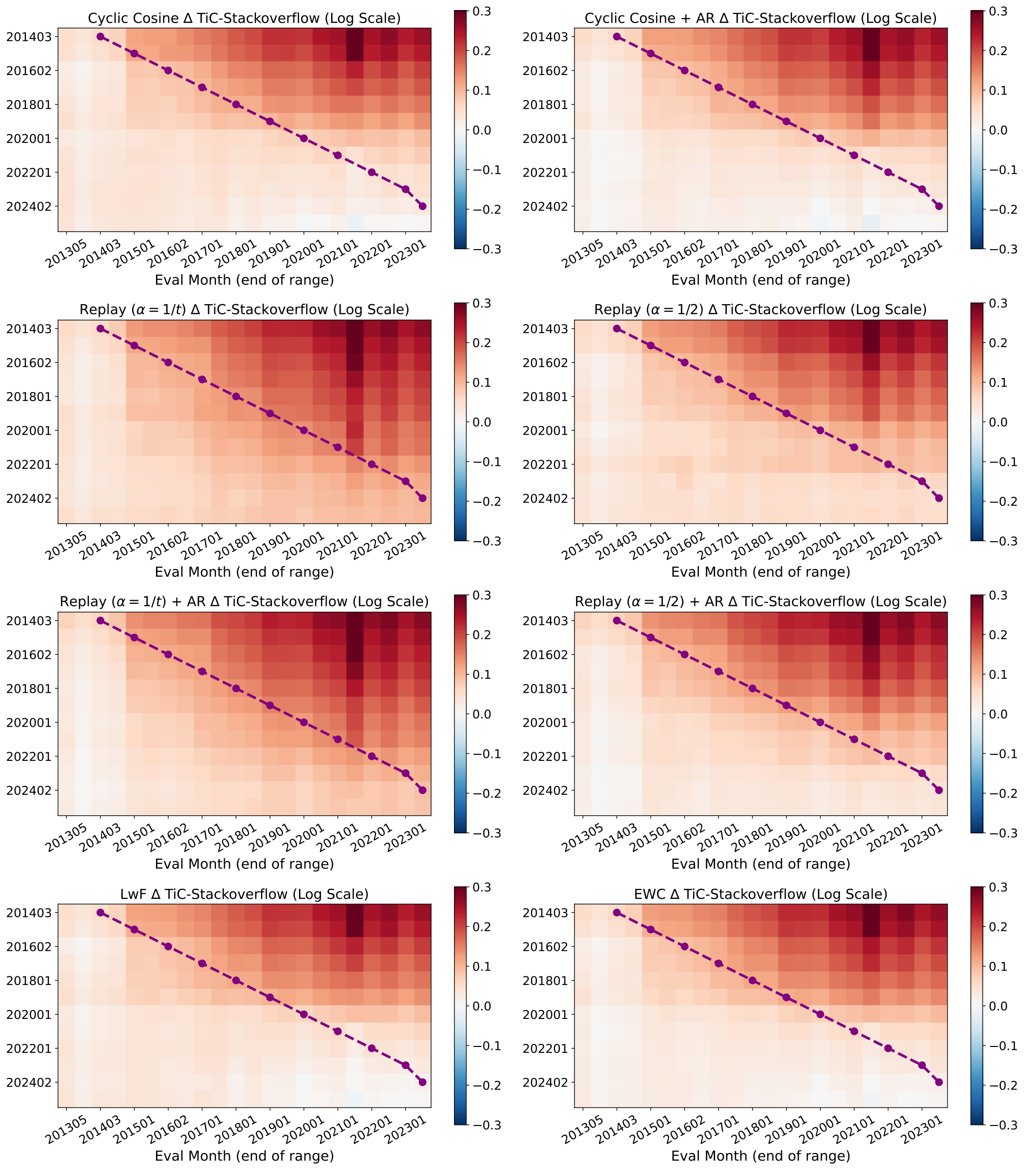


Figure 12: Heatmaps for various methods on the StackOverflow site of TIC-STACKE.

Table 4: Average over an extended set of TIC-STACKE evaluations that we refer to as TIC-STACKE-CAT7. This includes the following sites: apple, codereview, electronics, english, gaming, math, and worldbuilding. Overall, we find that a combination of replay and AR meta-schedules does the most to reduce forgetting while EWC performs best on ID and Forward evaluations.

Method	**TIC-STACKE-CAT7↓**		
	Backward	ID	Forward
Cyclic Cosine (std)	0.045 (0.001)	0.050 (0.001)	0.071 (0.000)
Cyclic Cosine + AR	0.035	**0.044**	0.068
Replay ($\alpha_t = 1/t$)	0.036	0.052	0.072
Replay ($\alpha_t = 1/2$)	0.038	0.049	0.070
Replay ($\alpha_t = 1/t$) + AR	**0.031**	0.050	0.072
Replay ($\alpha_t = 1/2$) + AR	**0.032**	0.046	0.071
LwF	0.044	0.048	0.070
EWC	0.035	**0.043**	**0.067**

E Extended Related Work

Temporal knowledge evaluations. Language models are expected to have an understanding of time to answer questions about specific time periods and generally reason about time. Various benchmarks have been proposed to evaluate temporal knowledge of LLMs. TemporalWiki [20] evaluates the capability of models to update factual knowledge. TemporalWiki is constructed from the difference between four consecutive snapshots of Wikipedia and Wikidata. Our TIC-WIKI evaluation expands and improves on TemporalWiki in various ways (see Appx. C.4). StreamingQA [31] consists of human written and generated questions from 14 years of news articles. The evaluation is either open-book where a model receives a collection of news articles that contain the answer, or closed-book where the model is first fine-tuned on the training set containing the documents and then tested. We evaluate our TiC checkpoints on StreamingQA both in open/closed-book setups and find that there is high ambiguity in the questions that evaluates reasoning more than temporal knowledge understanding. TempEL [56] evaluates entity linking performance across 10 yearly snapshots of Wikipedia. Entity linking is the task of mapping anchor mentions to target entities that describe them in a knowledge base. In comparison, our TIC-WIKI evaluates general language and knowledge understanding. TempLAMA [9] constructs an evaluation for factual queries from Wikidata. They focus on temporally sensitive knowledge with known start and end dates in a specific Wikidata snapshot. Notably, they propose TempoT5 to jointly model text and timestamp which allows a language model to answer temporal questions that change over time such "Who is the president". EvolvingQA [25] is also a benchmark for training and evaluating on Wikipedia over time where a LLM automatically generates question-answers from 6 months of articles in 2023. We avoid using any LLMs for generating our evaluations to prevent transfer of bias. TIQ [22] benchmark consists of 10k questions-answers based on significant events for the years 1801–2025.

Temporal generalization. Beyond understanding the past, LLMs need to be prepared for the future. Li et al. [29] observes performance deterioration of public LLMs on Wikipedia, news, code, and arXiv papers after their training data cutoff date. They particularly use compression rate achieved by treating an LLM as a general input compressor using arithmetic coding [8]. Our comprehensive evaluations on CommonCrawl, Wikipedia, news articles, StackExchange, and code evaluations verifies their results and more comprehensively shows that the rate of deterioration is domain-specific. DyKnow [41] evaluations also reaffirm that LLMs private and open-source LLMs have outdated knowledge by asking them questions constructed using Wikidata. They also observe LLMs output inconsistent answers in response to prompt variations and current knowledge editing methods do not reduce outdatedness. TAQA [57] further demonstrate that pretrained LLMs mostly answer questions using knowledge from years before their pretraining cutoff. They construct question/answers from Wikipedia for years 2000–2023 and propose three methods to improve the temporal alignment of models. Similar observations have been made in RealTimeQA [24] and TempUN [4]. These works further solidify the need for continuously updating models with continual pretraining.

Temporal understanding. General temporal understanding involves reasoning based on the relation between existing knowledge. Test of Time [12] benchmark evaluates temporal reasoning, logic, and arithmetics by constructing a synthetic dataset. Their goal is to reduce the chance of factual inconsistency in the evaluation using synthetic data. Our benchmark is designed to be fully realistic based on real data and timestamps to understand the challenges of large-scale continual pretraining in practice. Gurnee and Tegmark [15] find that LLMs learn a representation of space and time with individual neurons that encode spatial and temporal coordinates. They construct datasets of named entities and find that linear probing LLMs performs well on predicting spatial and temporal coordinates. Nylund et al. [42] proposed time vectors that specify a direction in the model's weight space that improve performance on text from a specific time period.

Temporal domain-specific evaluations. We can further analyze the temporal understanding of a model based on the performance on specific domains with varying rates of change. Luu et al. [35] studied temporal misalignment such as quantifying temporal degradation of domain-specific finetuning in four domains: social media, science, news, and food reviews. They observed significant temporal degradation in domains such as news, social media, and science but less in food reviews. Gururangan et al. [16] studied domain-adaptive pretraining and task-adaptive pretraining on unlabeled data for four domains in science, news, and reviews. They observe domain/task-adaptive pretraining improves performance on the new domain but do not evaluate forgetting on previous domains. Agarwal and Nenkova [1] studies the temporal model deterioration on future evaluations. They find that the deterioration is task-dependent and domain-adaptive pretraining does not help hypothesizing that limited pretraining data is detrimental in continual pretraining. Jin et al. [23] domain-incremental pretraining for four scientific domains as well as temporal pretraining on social media over 6 years. They focus on the impact on downstream performance after fine-tuning. They observe distillation-based approaches are the most effective in retaining dowstream performance for tasks related to earlier domains. Overall, the gap between different continual learning methods remained small that can be due to the small scale of pretraining. In comparison, our TiC-CC training is simulating large-scale pretraining.

Domain/task-continual learning for LLMs. In domain/task continual learning, the model is presented with a sequence of tasks with predefined labels [18, 52, 59]. Each task comes with its training and test sets. In contrast with continual pretraining, the model needs to support a growing set of labels while compared with temporal continual learning, the order of tasks are often arbitrary (e.g., Split-CIFAR, Perm-MNIST). Prominent methods in this domain are regularization-based methods [26, 37, 38, 11], replay-based methods that often perform superior [32, 2, 45], and architecture-based methods that adapt the model over time [51, 50]. Continual learning for language models has also been dominated by domain/task continual works. Jin et al. [23] proposed benchmarks for continually training models on a sequence of research paper domains as well as chronologically-ordered tweet streams. Razdaibiedina et al. [47] proposed learning a new soft prompt for each task and pass soft prompts for all seen tasks to the model which provides adaptability while preventing catastrophic forgetting. Luo et al. [34] studied continual learning for instruction tuning and observed catastrophic forgetting, especially for larger models. Mehta et al. [36] showed that generic pretraining implicitly reduces catastrophic forgetting during task incremental finetuning.

Continual pretraining of LLMs. Recent work have studied continual pretraining of foundation models at large-scale. TiC-CLIP [13] proposed a benchmark of training and evaluation of image-text foundation models and demonstrated the deterioration of existing foundation models on new data. Lazaridou et al. [27] studied time-stratified language pretraining on WMT, news, and arXiv up to 2019 and observed the models become outdated quickly on news data that holds even for models of various sizes. They study dynamic evaluation as a continual pretraining method that trains on a stream of chronologically ordered documents and observed that models can be updated. However, they did not explore the impact on forgetting and scalability of the method to more generic pretraining data over years. Jang et al. [21] proposed continual knowledge learning as a new problem and suggested that parameter expansion is necessary to retain and learn knowledge. They focus on one-step continual pretraining where models are pretrained on C4/Wikipedia data up to 2020 and then trained once more on recent news articles. They find adapter methods perform better than regularization and replay methods. Adapter methods are not directly applicable in our multi-year continual pretraining setup where we train in more than 100 steps on large-scale data. Gupta et al. [14] proposed simple recipes for continual pretraining of LLMs such as utilizing cyclical learning rate schedules with warmup and ablated on hyperparameters such as warmup duration when continuing the pretraining on a fixed pair of pretraining datasets.

Time-aware training. Orthogonal to continual pretraining, one can modify the training or fine-tuning of a model to include explicit information about time. TempLAMA [9] proposed prepending a time prefix to each example during training which gives the model the flexibility to respond to time-sensitive questions. They train models on news articles where the time can be reliably extracted. Drinkall et al. [10] proposed training a series of models with sequential data cutoffs dates to avoid data contamination with benchmark and private data. The observe no difference across time on static downstream evaluations when training models on news and Wikipedia

Factual editing and retrieval augmented generation (RAG). Another set of works aims to address the staleness of pretrained LLMs without further standard pretraining. One approach is to surgically edit the facts a model "knows" by identifying and updating the relevant weights within a model [39]. Another is to store edits in an explicit memory and learn to reason over them [40]. Retrieval augmented generation (RAG) pairs an LLM with new data sources to retrieve the most relevant document for a query. Generally, continual pretraining and RAG are orthogonal approaches to generate up to date responses. RAG methods increase the cost at inference time without changing the model while continual pretraining is the opposite. FreshLLMs [53] proposes a QA benchmark and argues that fast-changing knowledge requires a retrieval-based solution compared with slow-changing knowledge. Continual pretraining can be crucial in reducing the cost of RAG by utilizing retrieval only on knowledge that changes faster than the rate of continual pretraining.

F Future Work

Tokenizer. As the data changes over the years, new words appear in the language that would benefit from temporal adaptation of the tokenizer [58]. In this work, we fixed the tokenizer and did not change it across models. One important challenge that changing the tokenizer introduces is that the perplexity of models with different vocabularies will not be directly comparable. Future work would need to either focus on non-perplexity evaluations [8] or normalize perplexity by a mapping between vocabularies of a checkpoint to the reference oracle model.

Joint training of text and timestamp. TIC-CC training data has monthly timestamp corresponding to the crawl time that could be passed as context to the LLM during training and evaluation. TempoT5 [9] and TempoBERT [48] explored temporal language modeling for example by prefixing the input with "Year: " which helps resolve ambiguity in knowledge that has time-dependent answers such as "Who is the president".