TemporalWiki: A Lifelong Benchmark for Training and Evaluating Ever-Evolving Language Models

Joel Jang[1,*] **Seonghyeon Ye**[1,*] **Changho Lee**[3] **Sohee Yang**[1]
Joongbo Shin[2] **Janghoon Han**[2] **Gyeonghun Kim**[2] **Minjoon Seo**[1]
[1]KAIST AI [2]LG AI Research [3]Korea University
{joeljang,vano1205,sohee.yang,minjoon}@kaist.ac.kr ckdgh0801@korea.ac.kr
{jb.shin,janghoon.han,ghkayne.kim,stanleyjk.choi}@lgresearch.ai

Abstract

Language Models (LMs) become outdated as the world changes; they often fail to perform tasks requiring recent factual information which was absent or different during training, a phenomenon called *temporal misalignment*. This is especially a challenging problem because the research community still lacks a coherent dataset for assessing the adaptability of LMs to frequently-updated knowledge corpus such as Wikipedia. To this end, we introduce TEMPORALWIKI, a lifelong benchmark for ever-evolving LMs that utilizes the difference between the consecutive snapshots of English Wikipedia and English Wikidata for training and evaluation, respectively. The benchmark hence allows one to periodically track an LM's ability to retain previous knowledge and acquire updated/new knowledge at each point in time. We also find that training an LM on the *diff* data through continual learning methods achieves similar or better perplexity than on the entire snapshot in our benchmark with 12 times less computational cost, which verifies that factual knowledge in LMs can be safely updated with minimal training data via continual learning. The dataset and the code will be available at www.omitted.link.

1 Introduction

Large Language Models (LMs) pretrained on a vast amount of text corpus have shown to be highly effective finetuned or prompted to perform various downstream tasks (Raffel et al., 2019; Brown et al., 2020; Sanh et al., 2021; Wei et al., 2021). However, most of the datasets used to evaluate these LMs are static benchmarks; the train and test data are both from similar points in time. On the other hand, in the real world, factual knowledge is frequently changed, added, or deprecated. For example, suppose a language model is asked what the most dominant coronavirus variant is (Figure 1). The answer would have been the *Delta variant* in the fall of 2021 but has changed to the *Omicron variant* near the end of 2021. If LMs remain unchanged and are not periodically trained to cope with the changing world, they will be outdated very quickly. This means downstream tasks that directly depend on or are finetuned from the LM will suffer from *temporal misalignment* (Luu et al., 2021; Lazaridou et al., 2021), which refers to the misalignment in time between the train and test data.

Temporal misalignment becomes a critical problem, especially when using language models for knowledge-intensive tasks such as closed-book question answering (Roberts et al., 2020; Petroni et al., 2021; Jang et al., 2021) since they rely solely on the knowledge stored in their parameters. Furthermore, LMs augmented with retrieval mechanism (Guu et al., 2020; Lewis et al., 2020; Borgeaud et al., 2021) often suffer from *hallucination* even if they successfully retrieve up-to-date information (Zhang and Choi, 2021; Chen et al., 2021; Longpre et al., 2021). This means that the implicit knowledge stored in the model parameters has to be updated as well because it may cause conflicts with the explicit knowledge retrieved from external sources such as up-to-date knowledge bases and ultimately cause the LM to *hallucinate*.

Recently, Lazaridou et al. (2021); Jang et al. (2021) have explored updating the internal knowledge of LMs through continual pretraining on new and updated data as a solution for mitigating temporal misalignment. However, these datasets are still *static* in nature: as the world changes, they will eventually get outdated as well. In order to comprehensively measure the capability of ever-evolving LMs on addressing temporal misalignment, automated periodic evaluation of the LMs is crucial.

In this paper, we introduce TEMPORALWIKI, a *lifelong* benchmark for training and evaluating ever-evolving LMs in a periodic and automated manner, shown in Figure 1. The corpora used for updating

[*] indicates equal contribution.

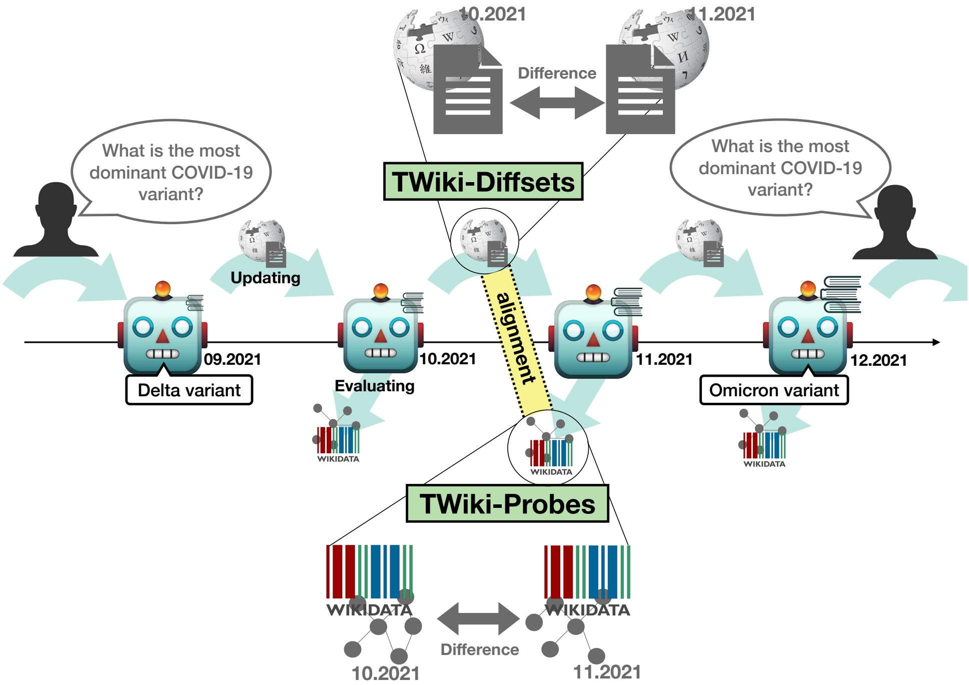


Figure 1: An overview of using TEMPORALWIKI, consisting of TWIKI-DIFFSETS and TWIKI-PROBES to train and evaluate ever-evolving LMs, respectively. Differences between Wikipedia snapshots at different points in time are used for temporal language modeling, and categorized factual instances in the corresponding Wikidata snapshots are used for temporal evaluation.

LMs are constructed by comparing articles from consecutive English Wikipedia snapshots and retrieving only *changed* information, which we name as TWIKI-DIFFSETS. The evaluation datasets are constructed in a similar manner by comparing English Wikidata snapshots that correspond to the Wikipedia snapshots in time and categorizing each factual instance into UNCHANGED or CHANGED. Since Wikidata updates may not exactly align with Wikipedia updates, we only retain factual instances that can be grounded to articles in Wikipedia, ensuring the quality of the data and name the resulting evaluation dataset as TWIKI-PROBES. The entire benchmark creation process is done without any human annotation, thus allowing it to be automated and *lifelong* as new English Wikipedia and English Wikidata snapshots are released by Wikimedia[1] on a monthly basis.

Through TEMPORALWIKI, we aim to tackle the following research questions: How can we train ever-evolving LMs efficiently and automate the evaluation of each update? How does updating LMs only on updated data from Wikipedia compare to updating LMs on entire Wikipedia snapshots, especially in scenarios with multiple updates? How problematic is catastrophic forgetting (McCloskey and Cohen, 1989) when LMs are updated only on updated data, and how can we effectively mitigate catastrophic forgetting? Our main contributions are summarized as follows:

- We introduce TEMPORALWIKI, a *lifelong* benchmark for ever-evolving LMs. Unlike previous *static* benchmarks, TEMPORALWIKI is responsive to the *dynamic* changes in the world and can be utilized to automatically train and evaluate ever-evolving LMs on each English Wikipedia and English Wikidata snapshot update.

- We find that continually training LMs only on the updated portion of English Wikipedia, which we call *temporal language modeling*, is much more computationally efficient than updating LMs on entire English Wikipedia snapshots as well as being more effective in terms of stability-plasticity trade-off. It is still a challenging task, especially when multiple updates are required due to catastrophic forgetting.

- As competitive baselines for temporal language modeling, we implement previous continual learning approaches that mitigate forgetting while bolstering the learning of new knowledge, thus providing an overall enhancement in terms of both stability and plasticity. We hope that TEMPORALWIKI will foster fu-

[1] https://commons.wikimedia.org/

ture research on continual learning methods for the temporal aspect of ever-evolving LMs.

2 Background

Recent works have introduced the need to tackle the issue of temporal misalignment, which refers to neural networks showing poor performance due to misalignment in time between the train and test data. Temporal misalignment can be caused either by (1) the dynamic nature of language (Röttger and Pierrehumbert, 2021; Hombaiah et al., 2021; Rosin et al., 2021; Loureiro et al., 2022) or (2) the update of factual information (Chen et al., 2021; Dhingra et al., 2021; Jang et al., 2021).

Luu et al. (2021) have emphasized the effect of temporal misalignment on eight different NLP downstream tasks, asserting that misalignment between the train and test sets of the downstream tasks causes severe performance degradation that can be mitigate finetuning on the corpus from the target period. Agarwal and Nenkova (2021) have argued this to be less of a concern when utilizing representations from pretrained LMs and show that self-labeling on the downstream task is more effective than continued pretraining on more recent data for temporal adaptation. Note that these works have focused on misalignment caused by the dynamic nature of language on tasks that are not knowledge-intensive, such as text classification.

Others have tackled the problem of temporal misalignment caused by the update of factual knowledge. Lazaridou et al. (2021) have shown that LMs deteriorate significantly in performance when there is a misalignment in time between the pretraining data and the downstream task and argued ever-evolving LMs are necessary. Dhingra et al. (2021) have proposed explicitly including time information during pretraining as a potential solution. Jang et al. (2021); Jin et al. (2021) have implemented continual learning methods to mitigate catastrophic forgetting that occurs during continued pretraining on new data.

Despite the recent surge of community interest in the need for ever-evolving LMs, the community still lacks widely-available resources to train and evaluate such LMs. Previous works have introduced benchmarks comprised of data sources from Twitter feeds (Osborne et al., 2014; Yogatama et al., 2014; Loureiro et al., 2022), recent news articles (Jang et al., 2021), and arXiv papers (Lazaridou et al., 2021) where the temporal adaptability of LMs and the effectiveness of different methodologies of updating LMs can be evaluated. However, these data sources are domain-specific and inherently *static*.

On the other hand, Wikipedia and Wikidata are known to be great sources of general world knowledge and thus have been widely used by the community (Dinan et al., 2019; Thorne et al., 2018; Kwiatkowski et al., 2019; Piktus et al., 2021). 120K volunteer editors make 120 updates to the English Wikipedia per minute and add hundreds of new article entries every day (Logan IV et al., 2021)[2]. Even though every Wikipedia and Wikidata update may not correspond to an actual change in the *real* world (i.e., every Wikipedia update may not be a result of actual change of world), TEMPORALWIKI leverages the dynamic nature of Wikipedia and Wikidata to provide a *lifelong* benchmark for developing and maintaining ever-evolving LMs.

3 TemporalWiki

In this section, we delve into the process of creating TEMPORALWIKI, which is comprised of training corpora (TWIKI-DIFFSETS) and evaluation datasets (TWIKI-PROBES) constructed from comparing the consecutive snapshots of English Wikipedia and English Wikidata, respectively. For efficiency purposes, *English* is abbreviated when referring to English Wikipedia and English Wikidata throughout the paper. Moreover, we clarify that not all Wikipedia/Wikidata updates equate to actual updates of *world* knowledge. In Section 3.1, we first describe the process of constructing the training corpora from Wikipedia snapshots. Then in Section 3.2, we describe the process of generating the evaluation datasets from Wikidata snapshots. In Section 3.3, we describe the quality control applied to the evaluation datasets, including the alignment of instances from Wikidata with Wikipedia. Lastly, in Section 3.4, we briefly discuss the current limitations of TEMPORALWIKI.

3.1 Generating Corpora for Temporal Language Modeling from Wikipedia

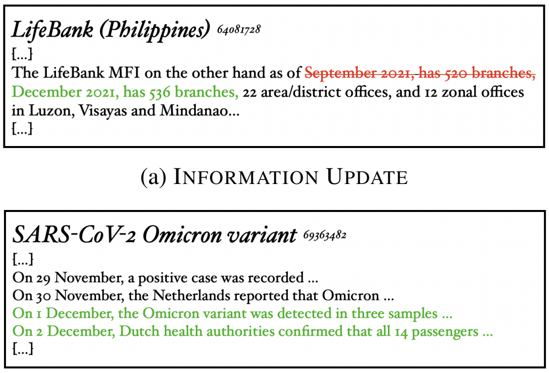
In terms of computational resources, it is highly inefficient to train an LM on the entire Wikipedia snapshot every time the LM requires updates since most part of Wikipedia is *unchanged* from the previous snapshot. Moreover, it is not certain whether updating the LM on the entire Wikipedia snapshot

[2] https://en.wikipedia.org/wiki/Wikipedia:Statistics

Algorithm 1 Generating TWIKI-DIFFSETS

```
Require: Wikipedia snapshots $WP_{prev}$ and $WP_{recent}$ where
  $WP_{recent}$ is more recent.
  $D$ := An empty array to store new and updated data.
  *$article$ in $WP$ has attributes $id$ and $text$
  for all $article\ a_r \in WP_{recent}$ do
    if $a_r.id = a_p.id$ for some $article\ a_p \in WP_{prev}$ then
      $D$.append(GETDIFF$(a_p, a_r)$)
    else
      $D$.append($a_r$)
    end if
  end for

  function GETDIFF$(a_p, a_r)$
  $Diff$ := An empty string to append difference between $text$
  in two $article$s.
  for all paragraph $p_r \in a_r.text$ do
    if $p_r$ have no matching sentences with any paragraph
    $p_p \in a_p.text$ then
      $Diff \leftarrow Diff + p_r$
    else if $p_r$ have some matching and some different sentences with any paragraph $p_p \in a_p.text$ then
      $Diff \leftarrow Diff$ + $sentences$ that differ between $p_r$ and
      $p_p$.
    end if
  end for
  return $Diff$
```



LifeBank (Philippines) 64081728
[...]
The LifeBank MFI on the other hand as of ~~September 2021, has 520 branches,~~ December 2021, has 536 branches, 22 area/district offices, and 12 zonal offices in Luzon, Visayas and Mindanao...
[...]

(a) INFORMATION UPDATE

SARS-CoV-2 Omicron variant 69363482
[...]
On 29 November, a positive case was recorded ...
On 30 November, the Netherlands reported that Omicron ...
On 1 December, the Omicron variant was detected in three samples ...
On 2 December, Dutch health authorities confirmed that all 14 passengers ...
[...]

(b) NEW INFORMATION

Figure 2: Examples of TWIKI-DIFFSETS constructed from comparing November 2021 and December 2021 Wikipedia Dumps. (a) shows an instance of information update and (b) shows an instance of new information.

is the best approach for updating the factual knowledge stored in the LM. Therefore, we compare the differences between consecutive Wikipedia snapshots in order to use only updated and new text for training. We call these subsets TWIKI-DIFFSETS. Algorithm 1 shows the procedure for generating them.

As shown in Algorithm 1, a single TWIKI-DIFFSET is generated by getting the differences (similarly to `git diff`) between two consecutive Wikipedia snapshots. If an article with a new unique id is included in the recent snapshot, we append the entire article to TWIKI-DIFFSET. For an article having an existing id in the previous snapshot, we compare the two articles by paragraphs and add new or updated sentences to TWIKI-DIFFSETS. Examples of TWIKI-DIFFSET are shown in Figure 2, and detailed Statistics are shown in Section 4.

3.2 Generating Evaluation Datasets from Wikidata

In this work, the main objective for continually pretraining LMs is to add and update the *factual* knowledge stored in the implicit parameters of LMs. The success of an LM update can be evaluated by quantifying the stability-plasticity dilemma (Mermillod et al., 2013): the dilemma of artificial and biological neural systems having to sacrifice either *stability*, ability to retain learned knowledge, or *plasticity*, ability to obtain new knowledge. In order to evaluate whether each update is successful, we need evaluation datasets that can quantify the amount of *changed* (updated or new) knowledge successfully gained (plasticity) and the amount of knowledge that remains *unchanged* as intended after the LM update (stability). Therefore, we categorize factual instances from Wikidata snapshots that are temporally aligned with Wikipedia snapshots and call the resulting datasets TWIKI-PROBES.

Wikidata snapshots are structured knowledge graphs that store factual information in the form of (`Subject`, `Relation`, `Object`) such as (`Barack Obama`, `born-in`, `Hawaii`). These factual instances can be used to probe the LM for factual knowledge (Petroni et al., 2019). Through Algorithm 2, we distinguish each factual instance into either UNCHANGED or CHANGED.

Algorithm 2 Generating TWIKI-PROBES

```
Require: Wikidata snapshots $WD_{prev}$ and $WD_{recent}$ where
  $WD_{recent}$ is more recent.
  $Un, C$ := Arrays that store UNCHANGED and CHANGED
  factual instances, respectively.
  for all fact $(s_r, r_r, o_r) \in WD_{recent}$ do
    $\mathbb{P} \leftarrow \{(s,r,o) \mid s = s_r$ where $(s,r,o) \in WD_{prev}\}$
    if $\mathbb{P} = \emptyset$ then
      $C$.append($s_r, r_r, o_r$)
    else if $r_r \notin \mathbb{P}$ then
      $C$.append($s_r, r_r, o_r$)
    else if $r = r_r$ and $o = o_r$ for some$(s,r,o) \in \mathbb{P}$ then
      $Un$.append($s_r, r_r, o_r$)
    else
      $C$.append($s_r, r_r, o_r$)
    end if
  end for
```

As shown in Algorithm 2, given two consecutive Wikidata snapshots, a single TWIKI-PROBE

Table 1: Examples of successful alignment between New factual instances from TWiki-Probes-0910 and articles from TWiki-Diffsets-0910. The alignment is considered successful because for the given factual instance, the `Subject` matches the title of the Wikipedia page and the `Object` exists in the article.

Subject	Relation	Object	Corresponding Sentence
Carlo Alighiero	place of death	Rome	[...] **Carlo Alighiero** died in **Rome** on 11 September 2021 at the age of 94.[...]
Shang-Chi and the Legend of the Ten Rings	instance of	Film	[...] **Shang-Chi and the Legend of the Ten Rings** is a 2021 American superhero **film** based on Marvel Comics featuring the character Shang-Chi.[...]
Out of Shadows	language of work or name	Spanish	[...] **It** was later translated into Portuguese, Turkish and **Spanish.**[...]
Mario Chalmers	member of sports team	Indios de Mayaguez	[...] On September 27, 2021, **Chalmers** signed with **Indios de Mayagüez** of the Baloncesto Superior Nacional.[...]

is constructed. The created TWiki-Probe is used to evaluate an LM updated with TWiki-Diffset, constructed with the two consecutive Wikipedia snapshots with the same timestamp. Algorithm 2 categorizes instances with new `Relation` or instances with the same `Relation`, but a new `Object` into Changed, and unchanged instances into Unchanged.

3.3 Quality Control for Evaluation Data

We apply several quality control steps to the categorized factual instances from Section 3.2 (Algorithm 2) to best represent the actual change of knowledge from the LM update.

Alignment with TWiki-Diffsets We ensure correct alignment of Changed factual instances with articles in TWiki-Diffsets and Unchanged factual instances with articles from the entire Wikipedia since Wikidata updates do not necessarily entail Wikipedia updates and vice versa. In order to do this, we take three steps. **Step #1**: We crawl information from each Wikipedia article page to find the mapping to the corresponding Wikidata entity id and store the information as a dictionary. **Step #2**: Then, for each factual instance from Changed, we check if the `Subject` id can be mapped to an article from TWiki-Diffsets using the dictionary of id mappings. Likewise, for each instance from Unchanged, we check if the `Subject` id can be mapped to an article from Wikipedia. **Step #3**: Lastly, for a successfully mapped factual instance from Step 2 (whether it is Changed or Unchanged), we finally keep the instances only if the `Object` exists in the text of the article.

Heuristic Filtering In addition to the alignment with TWiki-Diffsets, in order to further ensure the quality of the evaluation datasets, we apply three heuristic filtering rules to strengthen the quality of the data. **Rule #1**: We remove the instances where either Subject or Object is a substring of the other. **Rule #2**: We remove the instances where Object contains more than 5 words. **Rule #3**: We limit the proportion of single Subject by 1% of total, and Relation and Object by 5% of total. Table 1 shows some examples of TWiki-Probes after quality control.

3.4 Limitations of TemporalWiki

As mentioned at the beginning of this Section, each Wikipedia and Wikidata update does not ensure an actual update of *real-world* knowledge. For example, an addition of a new Wikipedia page does not necessarily mean that all the information on the new page is *new* world knowledge. Likewise, *existing* factual knowledge may be added to Wikidata because Wikipedia and Wikidata do not cover all of the world knowledge and may have some missing information about the world.

Moreover, one aspect that is not covered in this work is *knowledge deletion*. While maintaining Wikipedia and Wikidata, volunteer editors not only update or add new information but also *delete* information that is incorrect or misinformed. As removing the misinformation and bias stored in LMs is an important issue and necessary for truly ever-evolving LMs, future work should address this aspect utilizing *deleted* information from general knowledge sources such as Wikipedia.

4 Dataset Statistics

In this paper, we construct TemporalWiki from 08.2021 to 12.2021[3] and its statistics are discussed below.

Training Corpora Statistics Statistics of Wikipedia snapshots and TWiki-Diffsets are shown in Table 2. An interesting aspect of TWiki-Diffsets is that the amount of information being

[3] As new Wikipedia and Wikidata dumps are available on a monthly basis, we provide the source code for constructing new TWiki-Diffs and TWiki-Probes at www.omitted.link.

Table 2: Statistics of TWiki-Diffsets. The two digits indicate the month of the year 2021 that the Wikipedia snapshot was obtained from. The four digits for Wiki-Diffset indicate the months of the two snapshots being compared. For instance, TWiki-Diffset-0809 indicates the difference between August (08) and September (09).

	# of Articles	# of Tokens
Wikipedia-08	6.3M	4.6B
TWiki-Diffset-0809	306.4K	347.29M
Wikipedia-09	6.3M	4.6B
TWiki-Diffset-0910	299.2K	347.96M
Wikipedia-10	6.3M	4.7B
TWiki-Diffset-1011	301.1K	346.45M
Wikipedia-11	6.3M	4.6B
TWiki-Diffset-1112	328.9K	376.09M
Wikipedia-12	6.3M	4.7B

Table 3: Detailed Statistics of TWiki-Probes during construction. **Un** and **C** represents Unchanged and Changed factual instances, respectively.

	Initial Categorization		→	**Alignment**		→	**Heuristic Filtering**	
Month	**Un**	**C**		**Un**	**C**		**Un**	**C**
0809	514,017	1,209,272		10,133	2,329		6,935	1,776
0910	544,708	1,196,806		10,625	2,621		7,340	1,982
1011	460,228	1,572,778		10,544	1,742		7,313	1,358
1112	463,623	1,653,709		10,580	3,472		7,293	1,951

updated and added (i.e., number of tokens in each subset) is similar for each month.

Evaluation Dataset Statistics The statistics of TWiki-Probes from the initial categorization from Algorithm 2 and quality control are shown in Table 3[4].

For further analysis, we break down the entity types of `Subject` and `Object`, and observe a similar proportion of each entity category for each month of TWiki-Probes (Appendix A). We also show the distribution of the top 30 most frequent `Relation` of Unchanged and Changed (Appendix B).

5 Experiments with TemporalWiki

In this section, we train and evaluate ever-evolving LMs with TemporalWiki, which consists of TWiki-Diffsets and TWiki-Probes. Section 5.1 describes the experimental settings. Section 5.2 describes the baseline methodologies for updating LMs. Section 5.3 shows evaluation results on the training corpora. Section 5.4 presents the experimental results.

5.1 Experimental Settings

For our experiments, we continue pretraining GPT-2 Large (Radford et al., 2019) (774M parameters), which serves as our baseline language model (LM). We first compare the baseline performances between updating GPT-2 with TWiki-Diffsets and updating it with entire Wikipedia snapshots and evaluate each update using TWiki-Probes. We also implement continual learning methods from literature known for mitigating *catastrophic forgetting* that occurs when updating GPT-2 with only TWiki-Diffsets. Further detailed configuration of the experimental settings is provided in Appendix C.

5.2 Baseline Models

Here we describe the baseline methods used for training and evaluation, namely Initial, Full, Diff, RecAdam, Mix-review,K-Adapter, and LoRA as shown in Table 4 and 5.

Initial As the starting model checkpoint for all of the experiments, we first bring the initially pretrained GPT-2 from Radford et al. (2019) and continue pretraining it on the 08.2021 Wikipedia snapshot for four epochs in total (around 546K global steps) so that the initial GPT-2 used for all of the experiments is updated with the last two years of world knowledge. We denote this checkpoint as Initial, and it serves as the initial checkpoint for all of the other methods.

Full We start from Initial and continue pretraining it on the entire Wikipedia snapshot of each month in a sequential manner. For example, after training on the 09.2021 Wikipedia snapshot from Initial, we continue training it on the 10.2021 Wikipedia snapshot and move on to the next snapshot. We denote the resulting model as Full. We iterate through the training data only once, which corresponds to an average of 4.6 billion token updates (140K global steps) for each month.

Diff We start from Initial and continue pretraining it on TWiki-Diffsets in a sequential manner. We denote the resulting model as Diff. Similarly to Full, we iterate through the training data only once, which is an average of 347 million token updates (12K global steps) for each month.

RecAdam We implement a *regularization-based* continual learning method for training large LMs called RecAdam (Chen et al., 2020) which

[4] A single Wikidata snapshot is comprised of 93 million distinct entities, where there are around 30 facts for each entity which amounts to roughly *2.8 billion factual instances*. Since most instances from Algorithm 2 are categorized into Unchanged, we randomly sample 0.1% of the factual instances after applying Algorithm 2.

places a stronger independent assumption among the model parameters, overcoming the limitations of implementing traditional methods such as EWC (Kirkpatrick et al., 2017) for training large language models. We set the hyperparameters of the optimizer identical to the original implementation.

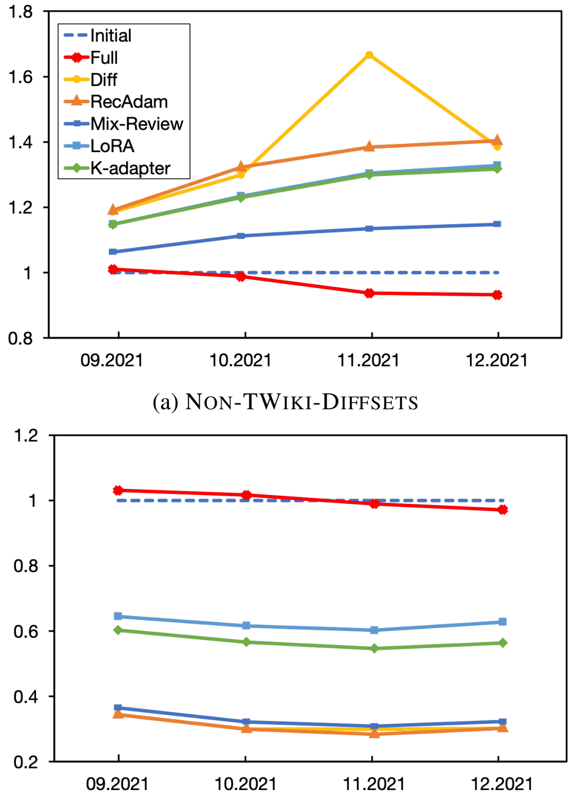
Mix-review We implement a *rehearsal-based* continual learning method for training large LMs called MIX-REVIEW (He et al., 2021) which mixes in random subsets of the initial pretraining data (08.2021 Wikipedia data). We fix the mix-ratio as 2 in our experiments.

LoRA We implement a parameter-expansion-based continual learning method called LORA (Hu et al., 2021) which freezes the original parameters while adding trainable rank-decomposition matrices into each layer. We use hyperparameters identical to the optimal setting of the original implementation.

K-Adapter We implement another parameter-expansion-based continual learning method called K-ADAPTER (Wang et al., 2021), which freezes the original parameters while adding additional adapters (an increase of 103M parameters) to the LM. [5]

5.3 Intrinsic Evaluation

We first perform intrinsic evaluation by measuring the perplexity of the baseline models on their training corpora. For each month, we measure the model's perplexity on TWIKI-DIFFSETS and NON-TWIKI-DIFFSETS, where the latter refers to the subset of the month's entire Wikipedia snapshot that does not include the data from TWIKI-DIFFSETS. We sample 10,000 input instances from each subset with a fixed length of 512 and measure the perplexity on proper noun tokens determined by a Part-of-Speech (POS) tagger (Honnibal and Montani, 2017) as in (Lazaridou et al., 2021), which can be considered as a proxy for tokens containing factual knowledge. Therefore, the result on NON-TWIKI-DIFFSETS is meant to indicate the performance on unchanged knowledge, while the result on TWIKI-DIFFSETS corresponds to updated and new knowledge. Figure 3 shows the relative perplexity of each baseline method compared to INITIAL (i.e., dividing each model by INITIAL, and thus the lower, the better).



(a) NON-TWIKI-DIFFSETS

(b) TWIKI-DIFFSETS

Figure 3: Relative proper noun perplexity of FULL, DIFF, and K-ADAPTER, LORA, RECADAM and MIX-REVIEW compared to INITIAL on TWIKI-DIFFSETS and NON-TWIKI-DIFFSETS for each month. Lower ratio indicates better performance. The performance of DIFF (orange) and RECADAM (yellow) in (b) is is almost identical.

Results on NON-TWIKI-DIFFSETS show that the relative perplexity of DIFF increases rapidly while that of FULL remains constant as time goes on, which implies that forgetting occurs when the LM is trained with TWIKI-DIFFSETS. The relative perplexities of continual learning methods increase less rapidly than DIFF, which means that applying continual learning mitigates catastrophic forgetting. MIX-REVIEW, especially, shows the least amount of forgetting among the continual learning methods, which indicates that training on the past corpus is effective in retaining performance on the previous training corpora in terms of *perplexity*.

On the other hand, the results on TWIKI-DIFFSETS show the opposite trend: the relative perplexity of DIFF is much lower than FULL. One thing to note is that the perplexity of FULL is very similar to that of INITIAL on TWIKI-DIFFSETS, which suggests that updating LMs on entire Wikipedia snapshots hinders the effective

[5] We add the additional parameters once for the updates from 08.2021. Exploring the optimal interval to add parameters for ever-evolving LMs is left for future work.

Table 4: Zero-shot perplexity of LMs measured on TWIKI-PROBES. **Time** represents the average training time of a single update under the setting described in Section 5.1. The description of each baseline model is explained in Section 5.2. Best performance is marked as **bold** while the second best is <u>underlined</u>.

		TWiki-Probes-0809			TWiki-Probes-0910			TWiki-Probes-1011			TWiki-Probes-1112		
	Time	**Un**	**C**	**Avg**	**Un**	**C**	**Avg**	**Un**	**C**	**Avg**	**Un**	**C**	**Avg**
INITIAL	0 hours	386.16	364.82	375.49	<u>356.66</u>	416.32	386.49	350.54	420.52	385.53	357.37	451.74	404.56
FULL	~24 hours	379.43	360.46	369.95	388.85	437.15	413.00	<u>337.34</u>	383.06	<u>360.20</u>	381.11	435.47	408.29
DIFF	~2.5 hours	409.31	284.34	346.83	409.86	<u>336.55</u>	373.21	465.20	367.72	416.46	391.77	365.07	378.42
RECADAM	~4 hours	358.10	**253.07**	**305.59**	376.12	**306.64**	<u>341.38</u>	439.14	<u>338.17</u>	388.66	400.56	<u>356.60</u>	378.58
MIX-REVIEW	~6 hours	**337.59**	<u>274.91</u>	<u>306.25</u>	394.20	381.21	387.71	375.85	369.50	372.68	**313.94**	**323.49**	**318.72**
LORA	~2 hours	386.52	332.98	359.75	359.54	371.03	365.29	381.80	391.66	386.73	361.42	408.19	384.81
K-ADAPTER	~2 hours	<u>340.47</u>	297.39	318.93	**326.53**	338.16	**332.35**	**325.11**	**332.61**	**328.86**	<u>333.53</u>	374.67	<u>354.10</u>

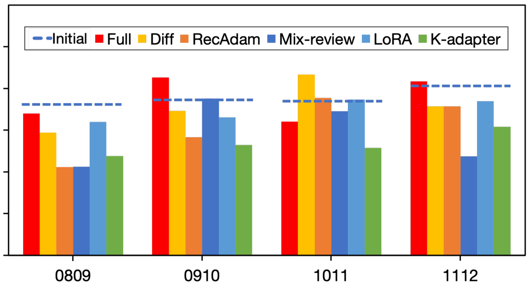


Figure 4: Average overall perplexity of TWIKI-PROBES. We average the perplexities of UNCHANGED and CHANGED with equal importance placed on stability and plasticity. The x-axis depicts the two-month intervals. A lower score indicates better performance.

learning of *changed* data compared to DIFF, despite both having seen the same instances of TWIKI-DIFFSETS during training for the same number of iterations. Among continual learning methods, K-ADAPTER and LORA shows higher overall perplexities than DIFF while MIX-REVIEW and RECADAM shows similar perplexity to DIFF on TWIKI-DIFFSETS.

5.4 Extrinsic Evaluation on TWIKI-PROBES

Performing only intrinsic evaluation on the training corpora is not sufficient because the intrinsic evaluation itself only tests the capability of the LMs for memorization (McCoy et al., 2021). Through extrinsic evaluation with TWIKI-PROBES (Section 3.2), we specifically focus on evaluating *factual* knowledge of the LMs from each update. Placing equal importance on *stability* (UNCHANGED) and *plasticity* (CHANGED), we show the average of the perplexities of UNCHANGED and CHANGED as well as individual perplexities in Table 4, and show a bar graph of the average perplexities in Figure 4[6].

As shown in Table 4, DIFF and all continual learning methods show better overall performance on CHANGED factual instances than INITIAL in all months, bolstering the results from the intrinsic evaluation. For UNCHANGED, however, DIFF suffers from *catastrophic forgetting*, showing consistent performance degradation as the number of updates increases. In contrast, continual learning methods effectively mitigate much of the catastrophic forgetting during temporal language modeling, resulting in lower perplexity on UNCHANGED, except RECADAM which performs worse as the number of updates increases. K-ADAPTER, especially, shows surprising results on UNCHANGED, outperforming even FULL throughout all of the months. Moreover, all continual learning methods surpass or are on par with DIFF on CHANGED factual instances, showing that continual learning methods do not hinder the LM from effectively learning new knowledge (plasticity).

Moreover, as shown in the average perplexity column of Table 4 and Figure 4, K-ADAPTER shows the most robust performance throughout the time periods. It is important to note that K-ADAPTER is around 12 times more computationally efficient than FULL in terms of total training time. DIFF also outperforms FULL in all months but 1011, showing that temporal language modeling itself is an effective approach for overall stability-plasticity trade-off.

We note that, as also shown in previous works (Lazaridou et al., 2021), results in Table 4 present an overall high perplexity (>200) because the sentences in TWIKI-PROBES are not natural sentences; they are factual phrases *synthetically* generated from a naive concatenation of `Subject`, `Relation`, and `Object`[7]. We address this issue

[6] The perplexity of UNCHANGED and CHANGED were each calculated by measuring the average perplexity of generating each factual instances.

[7] Using the pre-defined templates of LAMA (Petroni et al., 2019) seems to be an option, but we find that those templates do not fit well to our experiments because there is a considerable distribution gap between LAMA and TWIKI-PROBES; over half of the instances of TWIKI-PROBES are filtered out to apply the templates, especially for CHANGED.

via *light-tuning* in Appendix D.

Effect of Temporal Misalignment We quantify the effect of temporal misalignment on each method by training the LMs and evaluating their zero-shot perplexity on NEW instances of TWIKI-PROBES with various time intervals of training and evaluation. Among continual learning methods, we select K-ADAPTER since it shows the most robust performance for extrinsic evaluation across all time periods. As shown in Figure 5, FULL method is mostly influenced by the number of training updates and not much by whether there is temporal alignment. Since FULL is continuously pretrained on the entire Wikipedia corpus in each month, it would have likely seen the data containing CHANGED factual instances multiple times, leading to lower perplexity as training steps increases.[8] For DIFF and K-ADAPTER, there is a general trend of strong performance when there is temporal alignment (diagonal entries), outperforming FULL with much fewer global training steps. It is important to note that K-ADAPTER shows robustness against temporal misalignment, i.e., the perplexity does not increase much even when the training and evaluation months do not match, compared to DIFF which suffers from a more severe perplexity spike.

6 Conclusion

In this paper, we provide some answers to the four proposed questions in Section 1. (1) *How can we train ever-evolving LMs efficiently and automate the evaluation of each update?* We introduce TEMPORALWIKI, a lifelong benchmark that can be used for training and evaluating ever-evolving language models (LMs) in an automated manner. It consists of TWIKI-DIFFSETS as the training corpora for temporal language modeling and TWIKI-PROBES as the evaluation datasets for measuring the stability-plasticity trade-off at each LM update. (2) *How does updating LMs only on new and updated data from Wikipedia compare to updating LMs on entire Wikipedia snapshots, especially in scenarios with multiple updates?* Through experiments on TEMPORALWIKI, we show that updating LMs on TWIKI-DIFFSETS leads to better acquisition of *new* and *updated* knowledge than updating on entire Wikipedia snapshots with much less computational cost. (3) *How serious is catastrophic forgetting when LMs are updated only on new and updated data?* Temporal language modeling is still a challenging problem, as we observe more forgetting of previous knowledge not contained in TWIKI-DIFFSETS as the number of LM updates increases. However, results still show an overall enhancement in terms of stability and plasticity compared to updating with entire Wikipedia snapshots, showing that temporal language modeling can also be an effective alternative. (4) *How can we mitigate catastrophic forgetting?* We find that continual learning methods (regularization, rehearsal, and parameter-expansion) specific for large language model training effectively mitigates forgetting and shows robust performance in terms of balancing the overall trade-off between stability and plasticity on TWIKI-PROBES.

References

Oshin Agarwal and Ani Nenkova. 2021. Temporal effects on pre-trained models for language processing tasks. *arXiv preprint arXiv:2111.12790*.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, et al. 2021. Improving language models by retrieving from trillions of tokens. *arXiv preprint arXiv:2112.04426*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. In *NeurIPS*.

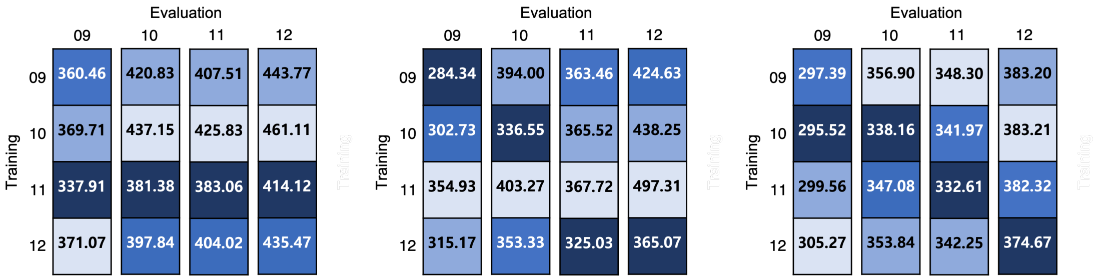
Sanyuan Chen, Yutai Hou, Yiming Cui, Wanxiang Che, Ting Liu, and Xiangzhan Yu. 2020. Recall and learn: Fine-tuning deep pretrained language models with less forgetting. In *EMNLP*.

Wenhu Chen, Xinyi Wang, and William Yang Wang. 2021. A dataset for answering time-sensitive questions. In *NeurIPS*.

Bhuwan Dhingra, Jeremy R Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W Cohen. 2021. Time-aware language models as temporal knowledge bases. *arXiv preprint arXiv:2106.15110*.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2019. Wizard of wikipedia: Knowledge-powered conversational agents. In *ICLR*.

[8] Although directly training INITIAL on the whole Wikipedia corpus of a specific month can be an alternative, we exclude it here because it would only learn the knowledge of the specific month and thus inappropriate for a truly ever-evolving setting.



(a) FULL (b) DIFF (c) K-ADAPTER

Figure 5: The zero-shot perplexity of the LMs updated and evaluated on various time intervals of CHANGED of TWIKI-PROBES, showing the effect of temporal misalignment. The better the results, the darker the performance is colored. The color is compared within the same method and also the same evaluation set.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. 2020. Realm: Retrieval-augmented language model pre-training. In *ICML*.

Tianxing He, Jun Liu, Kyunghyun Cho, Myle Ott, Bing Liu, James Glass, and Fuchun Peng. 2021. Analyzing the forgetting problem in pretrain-finetuning of open-domain dialogue response models. In *EACL*.

Spurthi Amba Hombaiah, Tao Chen, Mingyang Zhang, Michael Bendersky, and Marc Najork. 2021. Dynamic language models for continuously evolving content. In *KDD*.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2021. Towards continual knowledge learning of language models. *arXiv preprint arXiv:2110.03215*.

Xisen Jin, Dejiao Zhang, Henghui Zhu, Wei Xiao, Shang-Wen Li, Xiaokai Wei, Andrew Arnold, and Xiang Ren. 2021. Lifelong pretraining: Continually adapting language models to emerging corpora. *arXiv preprint arXiv:2110.08534*.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. 2019. Natural questions: a benchmark for question answering research. *TACL*, 7:453–466.

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. 2021. Mind the gap: Assessing temporal generalization in neural language models. In *NeurIPS*.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *NeurIPS*.

Patrick Lewis, Pontus Stenetorp, and Sebastian Riedel. 2021. Question and answer test-train overlap in open-domain question answering datasets. In *EACL*.

Robert L Logan IV, Alexandre Passos, Sameer Singh, and Ming-Wei Chang. 2021. Fruit: Faithfully reflecting updated information in text. *arXiv preprint arXiv:2112.08634*.

Shayne Longpre, Kartik Perisetla, Anthony Chen, Nikhil Ramesh, Chris DuBois, and Sameer Singh. 2021. Entity-based knowledge conflicts in question answering.

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. 2022. Timelms: Diachronic language models from twitter. *arXiv preprint arXiv:2202.03829*.

Kelvin Luu, Daniel Khashabi, Suchin Gururangan, Karishma Mandyam, and Noah A Smith. 2021. Time waits for no one! analysis and challenges of temporal misalignment. *arXiv preprint arXiv:2111.07408*.

Michael McCloskey and Neal J Cohen. 1989. Catastrophic interference in connectionist networks: The sequential learning problem. *Psychology of learning and motivation*.

R Thomas McCoy, Paul Smolensky, Tal Linzen, Jianfeng Gao, and Asli Celikyilmaz. 2021. How much do language models copy from their training data? evaluating linguistic novelty in text generation using raven. *arXiv preprint arXiv:2111.09509*.

Martial Mermillod, Aurélia Bugaiska, and Patrick Bonin. 2013. The stability-plasticity dilemma: Investigating the continuum from catastrophic forgetting to age-limited learning effects. *Frontiers in Psychology*.

Miles Osborne, Ashwin Lall, and Benjamin Van Durme. 2014. Exponential reservoir sampling for streaming language models. In *ACL*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, et al. 2021. Kilt: a benchmark for knowledge intensive language tasks. In *NAACL*.

Fabio Petroni, Tim Rocktäschel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, Alexander H Miller, and Sebastian Riedel. 2019. Language models as knowledge bases? In *EMNLP*.

Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Dmytro Okhonko, Samuel Broscheit, Gautier Izacard, Patrick Lewis, Barlas Oğuz, Edouard Grave, Wen-tau Yih, et al. 2021. The web is your oyster–knowledge-intensive nlp against a very large web corpus. *arXiv preprint arXiv:2112.09924*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. *JMLR*.

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *EMNLP*.

Guy D Rosin, Ido Guy, and Kira Radinsky. 2021. Time masking for temporal language models. *arXiv preprint arXiv:2110.06366*.

Paul Röttger and Janet B Pierrehumbert. 2021. Temporal adaptation of bert and performance on downstream document classification: Insights from social media. In *Findings of EMNLP*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Leslie N Smith. 2018. A disciplined approach to neural network hyper-parameters: Part 1–learning rate, batch size, momentum, and weight decay. In *CVPR*.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. Fever: a large-scale dataset for fact extraction and verification. In *NAACL*.

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Cuihong Cao, Daxin Jiang, Ming Zhou, et al. 2021. K-adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of ACL*.

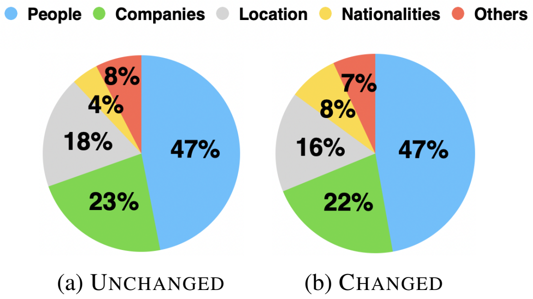
Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*.

Dani Yogatama, Chong Wang, Bryan R Routledge, Noah A Smith, and Eric P Xing. 2014. Dynamic language models for streaming text. *TACL*.

Michael JQ Zhang and Eunsol Choi. 2021. Situatedqa: Incorporating extra-linguistic contexts into qa. In *EMNLP*.

A Details of Entity Types of `Subject` and `Relation`

Figure 6 shows the ratio of different entity types of `Subject` and `Relation` of UNCHANGED and CHANGED.



(a) UNCHANGED (b) CHANGED

Figure 6: Entity types of `Subject` and `Object` in TWIKI-PROBES.

B Details of `Relation` Distribution

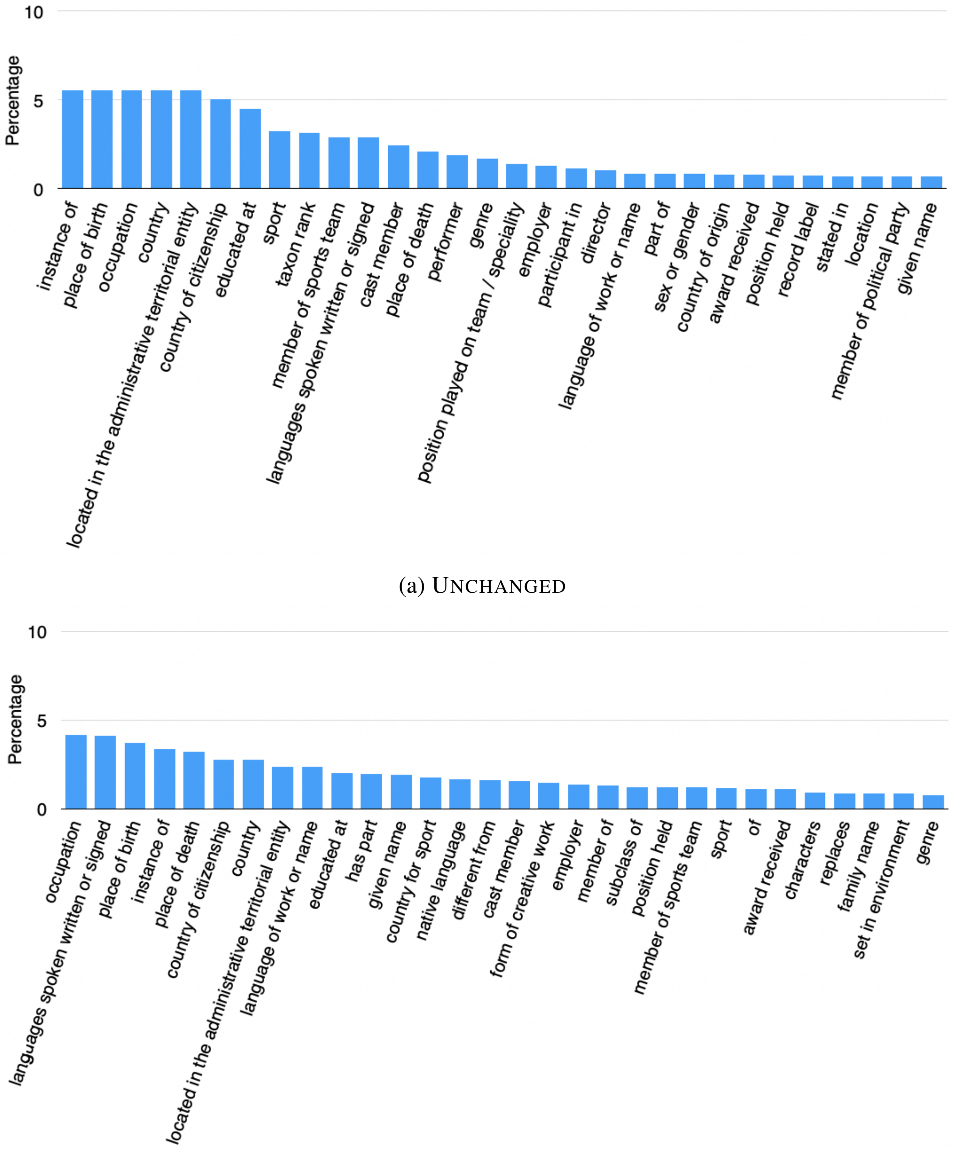
The distribution of `Relation` for UNCHANGED, CHANGED factual instances in TWIKI-PROBES is shown in Figure 7.

C Continual Pretraining and Light Tuning Configuration

For continual pretraining of LMs, we use 8 V100 GPUs with a global batch size of 64 and a fixed input sequence length of 512 for each update. We use the max learning rate of 1e-4 and one cycle learning rate scheduling policy (Smith, 2018). For light-tuning, the training is done for only one epoch with a learning rate of 1e-5 and a batch size of 32. Input and output sequence lengths are set to 25. For continual learning-based methods, we unfreeze all of the parameters during light-tuning, following Jang et al. (2021).

D Light-tuning results with TWIKI-PROBES

To alleviate the distributional shift that causes high zero-shot perplexity, we *light-tune* the LMs on 500



(a) UNCHANGED

(b) CHANGED

Figure 7: TWIKI-PROBES distribution of the top 30 `Relation`.

Table 5: Light-tuning perplexity of LMs measured on TWIKI-PROBES.

	TWiki-Probes-0809		TWiki-Probes-0910		TWiki-Probes-1011		TWiki-Probes-1112	
	Un	**C**	**Un**	**C**	**Un**	**C**	**Un**	**C**
INITIAL	116.99	142.58	**108.89**	167.82	106.14	172.18	**114.64**	177.02
FULL	124.37	145.89	112.51	172.70	**105.09**	164.59	118.54	164.17
DIFF	120.52	**116.44**	125.80	**142.82**	132.83	156.60	144.61	164.34
RECADAM	122.58	118.14	125.90	143.65	137.15	**148.24**	144.76	159.52
MIX-REVIEW	116.53	121.57	119.39	154.72	119.16	157.59	118.64	**145.29**
LORA	123.62	130.41	115.54	156.07	115.26	165.51	122.11	169.59
K-ADAPTER	**115.93**	134.46	116.27	154.11	110.17	158.21	117.22	167.44

instances randomly sampled from WikiData that do not overlap with instances from TWIKI-PROBES (details in Appendix E). Unlike finetuning, *light-tuning* lets the LM only learn the input and output distribution of the task, avoiding the problem of test-train overlap pointed out by Lewis et al. (2021). Table 5 shows the results of light-tuning, which demonstrate a similar trend as the zero-shot perfor-

Table 6: F1 score result of LMs on TWiki-Probes after light-tuning.

	TWiki-Probes-0809		TWiki-Probes-0910		TWiki-Probes-1011		TWiki-Probes-1112	
	Un	**C**	**Un**	**C**	**Un**	**C**	**Un**	**C**
Initial	6.98	3.19	7.26	3.35	7.27	2.74	6.84	2.82
Full	4.68	2.45	5.62	3.06	7.12	2.25	4.28	2.30
Diff	7.51	4.38	6.91	**4.46**	5.24	2.65	5.45	**4.38**
RecAdam	5.74	3.79	6.31	3.86	4.47	2.43	5.09	3.68
Mix-review	7.12	3.31	6.16	3.56	6.63	2.08	6.84	3.67
LoRA	7.36	**4.48**	7.23	3.89	7.19	3.87	6.82	3.81
K-Adapter	**7.54**	3.99	**7.34**	3.73	**7.38**	**3.91**	**6.87**	3.30

mance. Although light-tuning avoids the problem of test-train overlap, results are largely affected by the sampled instances for tuning, so a zero-shot evaluation setting is preferred for reliability.

Many knowledge-intensive tasks such as closed-book question answering (Roberts et al., 2020; Petroni et al., 2021; Jang et al., 2021) or slot filling (Petroni et al., 2021) use accuracy, EM, or F1 score to evaluate the task. We also show the F1 score on TWiki-Probes in Table 6. Overall trend is consistent with zero-shot perplexity metric; K-Adapter shows robust performance for both Unchanged and Changed.

E Light-Tuning Data

Table 7: Statistics of the data used for Light-Tuning

	Size	**# of Relation**	**Maximum Repetition of Relation**	**# of Subject**
Unchanged	500	102	58	499
Changed	500	140	31	500

We sample 5,000 instances from WikiData for each time step that do not overlap with instances from TWiki-Probes for each factual instance category. During sampling, we keep the distribution of each `Relation` proportional to the original distribution. Table 7 shows the size and distribution of `Relation` of light-tuning datasets.

eight