Evaluating the Ripple Effects of Knowledge Editing in Language Models

Roi Cohen[1] **Eden Biran**[1] **Ori Yoran**[1] **Amir Globerson**[1,2] **Mor Geva**[1,2,*]
[1]Blavatnik School of Computer Science, Tel Aviv University, Israel [2]Google Research, Israel
{roi1, edenbiran, oriy}@mail.tau.ac.il, {gamir, morgeva}@tauex.tau.ac.il

Abstract

Modern language models capture a large body of factual knowledge. However, some facts can be incorrectly induced or become obsolete over time, resulting in factually incorrect generations. This has led to the development of various editing methods that allow updating facts encoded by the model. Evaluation of these methods has primarily focused on testing whether an individual fact has been successfully injected, and if similar predictions for other subjects have not changed. Here we argue that such evaluation is limited, since injecting one fact (e.g., *''Jack Depp is the son of Johnny Depp''*) introduces a ''ripple effect'' in the form of additional facts that the model needs to update (e.g., *''Jack Depp is the sibling of Lily-Rose Depp''*). To address this, we propose novel evaluation criteria that consider the implications of an edit on related facts. Using these criteria, we then construct RippleEdits, a diagnostic benchmark of 5K factual edits, capturing various types of ripple effects. We evaluate prominent editing methods on RippleEdits, showing that they fail to introduce consistent changes in the model's knowledge. In addition, we find that a simple in-context editing baseline obtains the best scores on our benchmark, suggesting a promising research direction for model editing.[1]

1 Introduction

Modern language models (LMs) capture a large volume of factual knowledge in their parameters, which can be effectively utilized in downstream tasks (Petroni et al., 2019; Roberts et al., 2020; Shin et al., 2020; Razniewski et al., 2021; Heinzerling and Inui, 2021; Kadavath et al., 2022; Cohen et al., 2023). However, factual beliefs captured by the model may be incorrect or become outdated over time, potentially affecting the model's performance on downstream tasks, its reliability, and its usability (Dhingra et al., 2022; Lazaridou et al., 2021; Jang et al., 2022).

This limitation has prompted research on knowledge editing (KE) methods, which modify LMs to fix their factual errors (we provide a formal definition in §2). Knowledge editing work has focused on applying factual updates to LMs. Given an entity-relation-object triplet (e, r, o) representing a fact (e.g., *''Lionel Messi plays for the Inter Miami team''*), recent work proposed various methods (Mitchell et al., 2022; Meng et al., 2022, 2023; Hernandez et al., 2023b; Si et al., 2023) to inject this fact into the parameters of a given LM, while ''overriding'' beliefs the model might have on e and r (e.g., that Messi plays for Paris Saint-Germain).

A key question with KE is how to evaluate the success of such editing operations. The most basic ''sanity-check'' is that the model correctly completes $(e, r, ?)$, as well as other paraphrases of this task, with o. However, this is not enough as an evaluation, since one needs to check that the model did not distort other facts. Indeed, the standard evaluation protocol (Mitchell et al., 2022; Meng et al., 2022, 2023) for KE focuses on these two aspects of correctly completing various paraphrases of the new fact, as well as ensuring that other unrelated facts have not been changed.

In this work, we argue that to evaluate model edits, one should go beyond the single fact that was edited and check that other facts that are logically derived from the edit were also changed accordingly. For example, if z is the mother of e, then the children of z are the siblings of e. Consequently, once we modify the belief of a certain model that $z \rightarrow z'$ is the mother of e, then we should also ensure that the model's belief regarding the siblings of e is also correct. Figure 1 illustrates another example, where editing the `Team` for which `Lionel Messi` plays modifies other related facts, such as his country of residence, while other facts

*Work done at Google DeepMind.

[1]We release RippleEdits and our code at https://github.com/edenbiran/RippleEdits.

Transactions of the Association for Computational Linguistics, vol. 12, pp. 283–298, 2024. https://doi.org/10.1162/tacl_a_00643
Action Editor: Jing Jiang. Submission batch: 8/2023; Revision batch: 11/2023; Published 3/2024.
© 2024 Association for Computational Linguistics. Distributed under a CC-BY 4.0 license.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

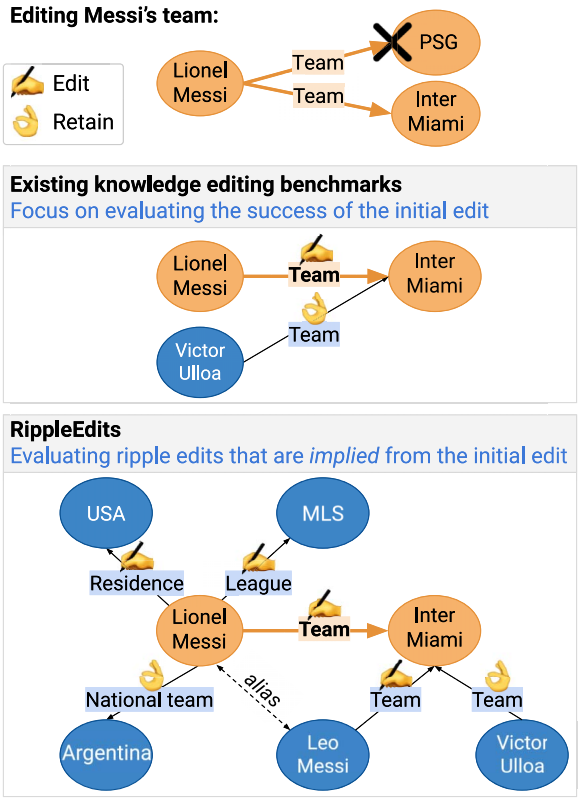


Figure 1: Illustration of the evaluation scope of RippleEdits, compared to existing knowledge editing benchmarks. For a given factual edit, we consider the ''ripple effect'' of the edit on the model's knowledge.

should be retained. We refer to such changes that are implied by a factual edit as *ripple effects*.

To account for ripple effects in the evaluation of factual edits, we propose six concrete evaluation criteria (see §3, Figure 2), for testing which facts other than the edit itself should be modified or retained post-editing. Our tests evaluate how well the model integrates the edit with the rest of its knowledge, through queries that involve logical reasoning, complex composition of facts with the edit as an intermediate step, subject aliasing, and specificity across relations.

Building upon these criteria, we create RippleEdits, a new benchmark for comprehensive evaluation of KE of LMs (see §4). RippleEdits includes 5K entries, each consisting of a factual edit, along with a set of test queries that check if the edit was successful in terms of its ripple effect. Moreover, RippleEdits contains meta-data for each edit, including information about the time-stamp of the edit (i.e., recent versus old), and the popularity of the entities (i.e., head versus tail).

We use RippleEdits to evaluate three popular editing methods on five recent strong LMs (see §5). We find that, even though current KE methods are effective in modifying a particular fact, they often fail to capture the ripple effects entailed by that fact, and demonstrate poor performance on most of our evaluation criteria. Moreover, analyzing how editing performance varies across model sizes and entity frequencies, we find that (a) larger models handle ripple effects better, and (b) editing frequent entities results in more logical reasoning errors.

Last, we consider a simple in-context editing baseline for KE that leverages the casual attention mechanism rather than explicit parametric updates. While this method achieves the best results on our benchmark, outperforming current parametric KE methods, there is still ample room for improvement that calls for future research.

To conclude, our work makes multiple contributions: (a) it highlights key limitations of KE evaluation, specifically regarding ripple effects and introduces comprehensive evaluation criteria to mitigate those limitations, (b) it proposes RippleEdits, a benchmark inspired by these criteria, (c) it evaluates current methods for KE and shows that they do not perform well on this task, while demonstrating that in-context editing is a promising direction for KE. We release RippleEdits and our code to facilitate future work on KE.

2 Problem Setting

We consider editing of *factual knowledge*, where facts are expressed as triplets (e, r, o) of a subject entity e (e.g., `Lionel Messi`), a relation r (e.g., `Team`), and an object o (e.g., `Inter Miami`). We distinguish between two edit types, based on the knowledge encoded in the model before the edit: (a) *modification* of a fact that is already encoded in the model $(e, r, o) \rightarrow (e, r, o^*)$, that is, updating the object $o \rightarrow o^*$ for a given subject e and relation r, and (b) *injection* of a new fact (e, r, o^*) that is not captured by the model. Moreover, we note that for one-to-one relations like `Date of birth`, where there is a single object for a given subject, an injection edit can be viewed as populating an empty object $(e, r, \emptyset) \rightarrow (e, r, o^*)$. In contrast, for one-to-many relations, such as `Sibling` and `Occupation`, an injection edit augments the set of objects $(e, r,$

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

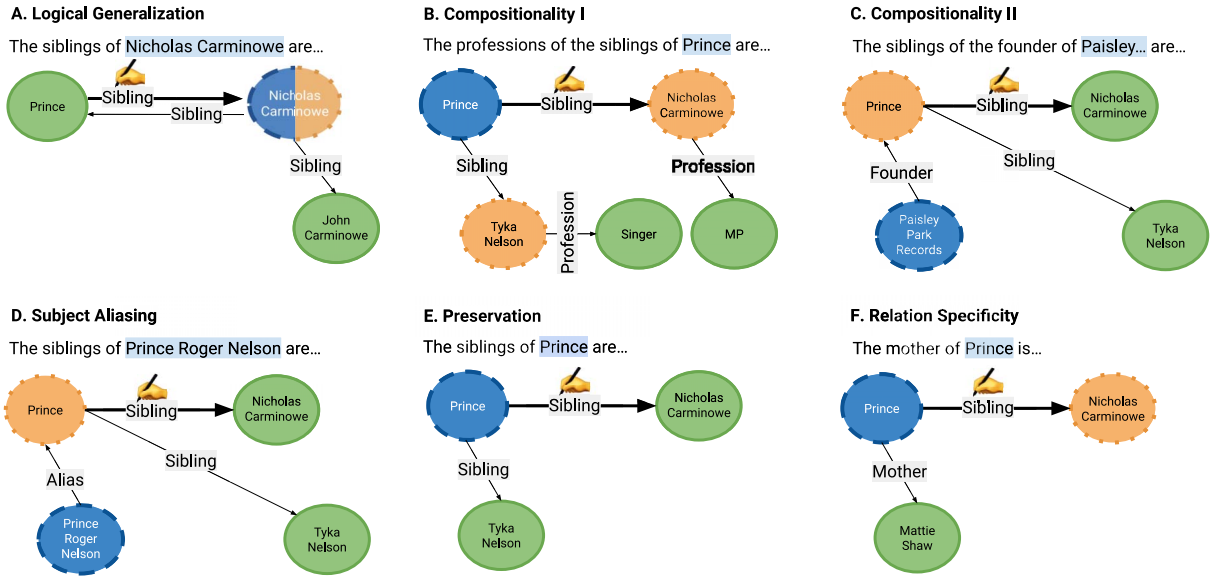


Figure 2: An illustration of our evaluation criteria, for an edit that simulates adding a sibling to the subject entity `Prince`, shown at the top of each graph with a **bold arrow** and an edit sign over the `Sibling` relation. For each criterion, the tested subject and target object are circles with **dashed blue line** and **solid green line**, respectively, and other nodes in **dotted orange line**. For *Logical Generalization* (A), the additional fact that needs to be inserted to the knowledge graph (KG) is presented with an edit sign next to the relation. We show the same node in different colors for completeness, as the tested subject is also the object in the edit that needs to be inserted. For *Compositionality I, II* (B, C), the model needs to hop over the edit to arrive at the target. In *Subject Aliasing* (D) we verify the edit also propagates to paraphrases of the input. In *Preservation* (E), we verify that other targets of the edited subject-relation are preserved. In *Relation Specificity*, we verify other relations for the subject are not modified.

$\{o_1, .., o_n\}) \rightarrow (e, r, \{o_1, .., o_n, o^*\})$. Whether an edit is viewed as a modification or injection, depends on whether that information was captured in the model before the edit. Moreover, evaluating if a specific fact (before or after an edit) is encoded by a model is typically done by testing if the model predicts the object for various input queries that represent the subject and relation (see more details in §3.2).

3 Ripple Effects of Factual Edits

We focus on evaluating the downstream effect of a given edit, i.e., given an edit $(e, r, o) \rightarrow (e, r, o')$, we expect certain facts related to the edit to change as well. Consider, for example, the edit shown in Figure 1. Changing the team for which Messi plays might also affect the league he plays in and his country of residence. Formally, for a given model, assume a knowledge-graph $\mathcal{K} := \{(e_i, r_i, o_i)\}_{i=1}^{N}$ of N factual triplets, representing the model's knowledge, and let $\delta : (e, r, o) \rightarrow (e, r, o')$ be an edit request for $\mathcal{K}$. We define the *ripple effect* of δ on $\mathcal{K}$, as the set of triplets $\mathcal{R}(\delta)$ that the model implicitly needs to inject, modify, or delete from $\mathcal{K}$ to reflect the world state after the edit.

Notably, different edits can cause ripple effects of varying magnitudes. For example, changing the country of Rome from Italy to France, will entail many follow-up changes, such as the country in which the Colosseum is located, the language spoken in Rome, and so forth. In contrast, updating the siblings of Prince (Figure 2) is both more realistic and should result in a more local effect. We refer to the number of facts affected by a single edit δ (i.e., $|\mathcal{R}(\delta)|$) as its *severity*. In general, editing popular entities that appeared frequently during training is likely to introduce more changes, and thus, editing their properties has a higher severity.

3.1 Evaluation Criteria

We wish to evaluate how well models capture the ripple effects of factual edits. However, given that ripple effects can potentially span a large number of implied edits, we focus on evaluating modified facts that are within a 2-hop distance from the

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

subject or object of the edit. Concretely, for an edit $\delta : (e, r, o) \rightarrow (e, r, o^*)$, we evaluate the ripple effect $\mathcal{R}(\delta)$, via the following evaluation criteria (examples are shown in Figure 2):

1. ***Logical Generalization*** **(LG)**: Relations in a knowledge graph satisfy certain logical constraints. For example, the relation `Sibling` is symmetric and therefore if $(e,$ `Sibling`$, o)$ is true then $(o,$ `Sibling`$, e)$ is also true, and vice versa (Figure 2A). Likewise, the relation `Location` is transitive so $(e,$ `Location`$, o) \wedge (o,$ `Location`$, z) \Rightarrow (e,$ `Location`$, z)$. We wish to check that such logical implications about the subject e, the original object o, and the new object o^*, hold after editing. We focus and elaborate on specific constraints in §4.

2. ***Compositionality I*** **(CI)**: As δ alters one edge in a knowledge graph, we can check the composition of this edge with other edges. Namely, we test if the model can compose the edited fact with other facts about the target object. Let (o, r', z) and (o^*, r', z^*) be two facts of the same relation about o and o^*, respectively. Also, denote by $r'' = r \circ r'$ the complex relation expressing the composition of r and r' (e.g., $r''=$ `Profession of sibling` for $r =$ `Sibling` and $r' =$ `Profession`). Then, after the edit δ, we expect the following change $(e, r'', z) \rightarrow (e, r'', z^*)$. For example (Figure 2B), the professions of the siblings of `Prince` can be modified once a new sibling is injected.

3. ***Compositionality II*** **(CII)**: We test if the model can compose the edited fact with facts about a different subject $e' \neq e$. Formally, let (e', r', e) be a fact about e' with e as its object, and denote by $r'' = r' \circ r$ the complex relation expressing the composition of r' and r (see an example in criterion 2). After the edit δ, the following change is expected for the subject e': $(e', r'', o) \rightarrow (e', r'', o^*)$. For instance (Figure 2C), changing the siblings of `Prince` also modifies the siblings of the founder of `Paisley Park Records` (i.e., r'' is a complex relation expressing ''siblings of the founder'').

4. ***Subject Aliasing*** **(SA)**: We test that editing a fact about e induces the same edit to other entities e' that are aliases of e, namely, $(e', r, o) \rightarrow (e', r, o^*)$. For instance (Figure 2D), modifying the siblings of `Prince` should also modify the sibling of his alias, `Prince Roger Nelson`.

5. ***Preservation*** **(PV)**: If r is a one-to-many relation, then adding a new object should not affect the other objects encoded about e. Hence, in such cases, we expect that any existing triplet (e, r, o') for an object $o' \neq o^*$ would remain following the edit. For example (Figure 2E), after inserting the sibling `Nicholas Carminowe` for `Prince`, the fact that `Tyka Nelson` is also his sibling should be retained.

6. ***Relation Specificity*** **(RS)**: We test that facts about e, with relations whose objects are not influenced by o, are indeed not affected by the edit. For example (Figure 2F), modifying the sibling of `Prince` should not change his `Mother`. Note that these facts complement those evaluated by *Logical Generalization*.

In §4.1, we describe how we generate factual editing evaluations, based on the above criteria.

3.2 Related Work

Knowledge Editing Methods Several methods have been proposed to edit the factual knowledge encoded in a model. De Cao et al. (2021) and Mitchell et al. (2022) suggested using hypernetworks to update the model weights. In addition, Meng et al. (2022, 2023) proposed to modify encoded facts by updating the weights of MLP layers, following recent observations that these layers can be cast as key-value memories (Geva et al., 2021) that store factual knowledge (Dai et al., 2022). Other methods learn encodings that update the hidden representations created during model inference (Hernandez et al., 2023a), or augment the input context with edits (Zhong et al., 2023; Zheng et al., 2023). In §5.1, we discuss state-of-the-art KE methods used in this work in greater detail.

Separately from factual KE, recent work has also studied how to inject new facts into a model. Previous methods suggested unsupervised pre-training (Roberts et al., 2020; Zhang et al., 2021), semi-parametric methods, where external information is added from a knowledge-base (Zhang

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

et al., 2019; Peters et al., 2019; Lewis et al., 2020; Zhang et al., 2022), using adapters to store knowledge (Wang et al., 2021a), or extending the MLP layers (Yao et al., 2022).

Knowledge Editing Evaluation Recently, there has been a growing interest in KE evaluation (Yao et al., 2023). The prominent benchmarks for evaluating factual KE are the Zero-Shot Relation Extraction (zsRE) (Levy et al., 2017; De Cao et al., 2021) and CounterFact (Meng et al., 2022). zsRE is a question-answering dataset for relation-specific queries, which includes human generated paraphrases that are used to measure robustness to semantically equivalent inputs. For example, for the triplet (`x`, `Country`, `y`), zsRE contains queries such as ''*In which country is x?*''. CounterFact offers a more challenging setting, where edits are counterfactuals of a low probability, such as changing the `City` of `The Louvre` from `Paris` to `Rome`.

Evaluation in zsRE and CounterFact focuses on three primary aspects of (a) *efficacy*: checking that the model generates the target object post-editing, (b) *paraphrasing*: testing robustness in generating the target for paraphrases of the input, and (c) *specificity*: verifying that facts not related to the edit are unaffected. In addition, CounterFact evaluates the generation quality of the edited model when prompted with the edit's subject, measuring: *consistency*, i.e., similarity with subjects that share the same property as the edited object, and *fluency* in terms of repetitiveness of the generated text. More broadly, previous work evaluated to which extent LMs have beliefs (Genin and Huber, 2022; Kassner et al., 2021; Hase et al., 2023), and Hase et al. (2023) examined if updating beliefs propagates to entailed facts, extending the Wikidata5m dataset (Wang et al., 2021b) to test editing specificity.

Recently, Onoe et al. (2023) introduce the task of *entity knowledge propagation*, aiming to examine the extent to which models are able to reason about emergent entities that did not appear in pretraining. In addition, Hoelscher-Obermaier et al. (2023) show that existing KE methods can have unwanted side effects and suffer from low specificity. A concurrent work by Zhong et al. (2023) introduces MQUAKE, a benchmark that tests the ability of models to perform multi-hop reasoning after edits. While each of these benchmarks focuses on a single consequence of editing, RippleEdits provides a general framework for evaluating various types of edit ripple effects. Last, Gupta et al. (2023) focus on editing commonsense knowledge and introduce MEMIT-CSKPROBE, a dataset for semantic generalization of commonsense edits. RippleEdits is different from MEMIT-CSKPROBE as it evaluates editing of factual knowledge rather than commonsense knowledge.

4 The RippleEdits Benchmark

In this section, we describe a data generation pipeline (§4.1) for factual edit requests and queries for evaluating their ripple effects. Then, we apply our pipeline to create the RippleEdits benchmark for comprehensive KE evaluation (§4.2), and validate the quality of the data (§4.3).

4.1 Data Generation Pipeline

We describe our data generation process (illustrated in Figure 3), that creates KE evaluation examples, each consisting of a factual edit request and a set of test queries that follow our criteria. Since the pipeline involves manual writing of templates and logical rules per relation, we restrict the edits and test queries to a fixed set of N_{rel} basic relations.[2]

Step 1: Factual Triplet Collection The first step of the pipeline (Figure 3A) is to collect facts, from which we will later create edit requests. To this end, we use WikiData, a relational knowledge base consisting of facts that are expressed as triplets (e, r, o), where e is a subject entity, r is a relation, and o is an object. We collect triplets of three types:

- **Recent**: To create ''real'' plausible edit requests, we collect triplets that were inserted to WikiData only recently, and represent relatively new facts. Therefore, they can be used to create injection edit requests for models that were trained before these facts were introduced, to simulate cases of an out-of-date model that requires factual updates. We collect such facts by randomly sampling triplets that have been modified during a range of 250 days after July 2022.

[2]The full list of relations is available in our codebase, example relations are shown in Figure 4.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

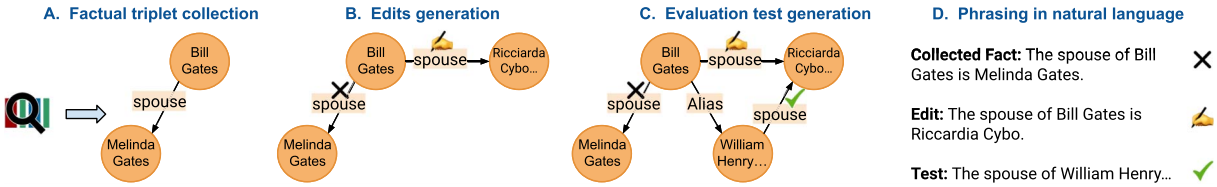


Figure 3: Illustration of our data generation process. We start by sampling a fact from a KG (A), here (`Bill Gates`, `Spouse`, `Melinda Gates`). Then, we generate the target triplet for the edit (B), in this case, choosing an object (`Ricciarda Cybo Malaspina`) that shares the same type as the original object. Next, we generate test queries (C) by sampling new triplets from the KG that should be retained or modified post-editing. Last, we utilize pre-defined templates to translate the KG triplets to natural language phrases (D).

- **RANDOM**: We collect triplets corresponding to random facts, for which we will later generate modification edits (similarly to Meng et al., 2022). These edits simulate factual edits that are meant to fix incorrect model predictions (e.g., predicting that the capital of Germany is Frankfurt). To this end, we divide the entities in WIKIDATA into 10 uniform buckets, based on the number of triplets associated with them. Intuitively, this can be viewed as a popularity measure. Then, we sample N_{ent} entities from each group and randomly choose one triplet for each entity.

- **POPULAR**: The two previous triplet types are randomly sampled from the entire knowledge base, and most of them are likely to represent facts about tail entities (except perhaps for a small subset in the top bucket). Such entities are often not captured by models (Mallen et al., 2023), and therefore not suitable for testing modification edits. To address this, we sample triplets from WIKIDATA with a subject that is a *popular entity*, namely, it appears in one of the top-viewed pages in Wikipedia.[3] Importantly, these types of triplets allow controlling for the ripple effect severity (§3), i.e., how models handle the ripple effects of popular entities versus tail entities.

Step 2: Edit Generation Once we obtain factual triplets, we turn to generate edit requests for them (Figure 3B). For RECENT, triplets represent new facts that are meant to be injected to the model, assuming that the latter was trained before these facts were introduced to the world. Hence, for RECENT, the target triplet for injection is the triplet itself.

For RANDOM and POPULAR triplets, we create an edit by generating a target triplet as follows. First, for every relation r, we create a set of candidate object entities O_r by sampling N_{cand} triplets $(e_1, r, o_1), \ldots, (e_{N_{cand}}, r, o_{N_{cand}})$ with the relation r, and extracting their objects $O_r = \{o_1, \ldots, o_{N_{cand}}\}$. Then, for every triplet (e, r, o) in RANDOM and POPULAR, we sample a target object $o' \neq o$ from O_r. Sampling the target object from triplets with the same relation makes the edit request technically consistent with the original triplet – the target object is of the same ''type'' as the original object (for example, a triplet with the relation `Capital` will get a new object of type `City`). The new triplet (e, r, o') will thus result in a ''fake'' fact, since it attaches a wrong object o' to the pair (e, r). For example, if RANDOM contains the triplet (`France`, `Capital`, `Paris`), its edit could be (`France`, `Capital`, `London`).

Step 3: Evaluation Test Generation The next step in the pipeline is to create ripple effect evaluations for the factual edits we collected (Figure 3C). To this end, we implement the evaluation criteria introduced in §3.1, and generate test queries for each criterion. Each test query corresponds to a triplet of subject and object entities and a possibly complex relation, that is expected to be true post-editing. In what follows, we provide details on our implementation, using objects from WIKIDATA.

For an entity e, we denote by $\mathcal{S}(e)$ the set of triplets in WIKIDATA in which e is the subject, and by $\mathcal{T}(e)$ the set of triplets in which e is the object. Moreover, for every relation r, we manually define a set D_r of relations that semantically depend on it. Namely, for a given subject, changing

[3] We extracted the entities whose corresponding Wikipedia page was included in the top-1000 most viewed pages in at least one month during 2020-2022.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

r's target object is expected to change the target objects for the relations D_r. For instance, the set D_r for the relation $r =$ `Mother`, includes the relations `Sibling`, `Sister`, `Brother`, `Aunt`, and `Uncle`, among others. Then, for every relation $r' \in D_r$, we craft a logical rule for obtaining the new target for that relation post-editing. For instance, for the relation $r =$ `Sibling`, we set a logical rule for $r' =$ `Mother` such that if (e, r, e') and (e', r', z') are true for entities e, e', z', then (e, r', z') should also be true.

Given an edit $(e, r, o) \rightarrow (e, r, o^*)$, we use D_r to generate test queries for *Logical Generalization* and *Relation Specificity*. For *Logical Generalization*, we apply the rule corresponding to each relation $r' \in D_r$ to obtain a set of test queries (x, r', z') about $x \in \{e, o, o^*\}$, where z' is the target obtained from the logical rule. For *Relation Specificity*, we create a test query for every triplet in $\mathcal{S}(e)$ with a relation that is *not* in D_r (but is in our set of N_{rel} relations).

To generate text queries for *Compositionality I*, we iterate through $\mathcal{S}(o^*)$ and for each triplet $(o^*, r', z) \in \mathcal{S}(o^*)$, we construct a two-hop query $(e, r \circ r', z)$ about e, with z as the answer. Similarly, for *Compositionality II*, we iterate through $\mathcal{T}(e)$ and for each triplet $(z, r', e) \in \mathcal{T}(e)$, we construct a two-hop query $(z, r' \circ r, o^*)$ about z with o^* as the answer. For *Subject Aliasing*, we use information maintained by WIKIDATA to create a test query (e', r, o^*) for every alias e' of e. Last, for *Preservation* we create test triplets $(e, r, o_1), \ldots, (e, r, o_n)$ that check if the model retained the original objects $\{o_1, \ldots, o_n\}$ in addition to the new edited object o^*.

Step 4: Phrasing in Natural Language At this point (Figure 3D), we have factual edit requests and their corresponding test queries. To use them as inputs to LMs, we convert them from triplet-form to natural language (NL). To this end, we manually craft a template NL phrase per relation (this is feasible since we use a fixed set of relations), and use it to convert all the triplets with this relation. For instance, the template ``The date of birth of <e> is'' converts triplets with the relation $r =$ `Date of Birth` and a subject entity e.

For the *Preservation* triplets generated for an edit $(e, r, \{o_1, \ldots, o_n\}) \rightarrow (e, r, \{o_1, \ldots, o_n, o^*\})$, where o^* is a new object added to a set of possibly multiple ($n \geq 0$) objects, we form a single NL query about other objects than the edited one, e.g., ``The award received by <e> which is not <o^*> is''.

	RECENT	RANDOM	POPULAR
# of factual edits	2,000	1,000	1,000
# of queries per edit	26.8	18.8	25.6
# of queries per criterion	5.24	3.1	4.2
# of LG queries	2.5	3.6	2.6
# of CI queries	11.7	4.7	6.1
# of CII queries	5.1	5.1	3.9
# of SA queries	1.8	1.3	4.7
# of PV queries	0.6	0.4	0.5
# of RS queries	5.1	3.7	7.8
Subject triplets count	31.7	13.3	115.2
Subject page back-links	278.1	121.6	3934.5
Subject page views	189.6	67.91	7376.5
Object triplets count	192.4	46.4	39.5
Object page back-links	18634.2	3065.0	2136.0
Object page views	2852.4	1379.7	1176.7

Table 1: Statistics per subset of RIPPLEEDITS, showing the average of different metrics. Breakdown by evaluation criteria shows the number of queries of each criterion per edit. For a given subject/object entity, triplets count is the number of WIKIDATA facts it is associated with, page back-links is the number of Wikipedia pages with a link to the entity's page, and page views is the recent average daily view count of the entity's page.

4.2 Data Statistics

We used our data generation pipeline to collect edits for 2,000 RECENT facts, 1,000 RANDOM facts, and 1,000 POPULAR facts, focusing on $N_{rel} = 54$ basic relations for which we manually crafted NL templates and logical rules.[4] To obtain the RANDOM subset, we set $N_{ent} = 200$ to sample 200 facts from each entity group in WIKIDATA. For edit generation of RANDOM and POPULAR, we set $N_{cand} = 100,000$. We call our diagnostic benchmark RIPPLEEDITS, and publicly release it to the research community. Notably, RIPPLEEDITS focuses on ripple edits and is meant to complement existing benchmarks, and so it does not include previous evaluations, such as subject specificity and model consistency.

Statistics on RIPPLEEDITS are presented in Table 1, showing that our generation process resulted in 18-26 test queries per edit and over 3 queries per evaluation test, on average. Moreover,

[4]We release the templates and rules in our codebase.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

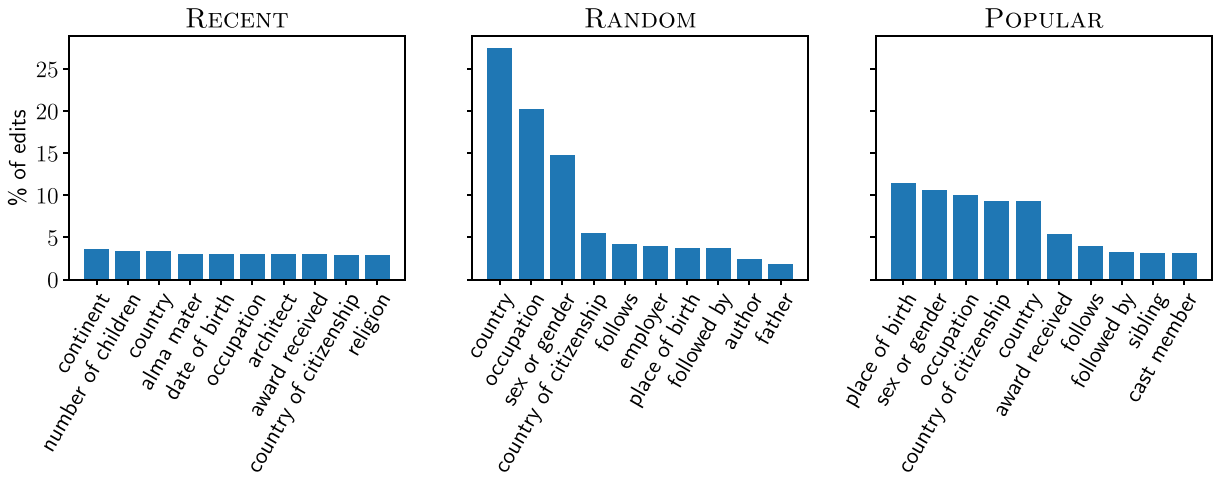


Figure 4: Most frequent relations and their frequency, in each subset of RIPPLEEDITS.

POPULAR edits contain more popular subjects (as intended), while RECENT edits have more popular objects. Figure 4 shows the top relations and their frequency in each subset of RIPPLEEDITS, demonstrating the diversity of the generated facts.

4.3 Data Quality

We conducted a manual analysis to validate that our generation pipeline produces valid test queries. Concretely, we sampled 200 random test queries from RIPPLEEDITS and checked the following two requirements: (a) *soundness*: the triplet that represents a given test query should be semantically correct, namely, the entity type of the object should match the relation type and the relation type should match the entity type of the subject. For example, queries such as *''The capital of Hilary Clinton is''* or *''The sibling of Lebron James is Los Angeles''* would have been disqualified. (b) *grammatically correct*: we check that the phrasing of the test query in natural language is grammatical.

We found that 100% of the queries were sound (i.e., semantically clear and correct), showing that the data curating process was designed properly. Furthermore, 98.5% of the queries were grammatically correct, while the ones which were not contain entity representations in a non-English language. This shows that our templates are general enough to properly fit various entity names.

5 Experiments

We use RIPPLEEDITS to evaluate recent KE methods, and show that despite substantial progress on existing benchmarks, current methods struggle to introduce consistent changes to the model's knowledge after an edit. Moreover, a simple in-context editing baseline that conditions the generation on the edited fact obtains better results, while leaving ample room for improvement for future research.

5.1 Evaluation Setting

Data To evaluate how well an editing method handles the ripple effects resulting from editing a given model, the data first needs to be adjusted such that (a) only cases of successful edits are evaluated, and (b) only test queries that the model answered correctly pre-editing are used for evaluation. Concretely, for an editing method $\mathcal{F}$ and a model $\mathcal{M}$, an edit request $x : (e, r, o) \rightarrow (e, r, o')$ is included in the evaluation if the following conditions are met when applying $\mathcal{F}$ to $\mathcal{M}$ and x: (a) $\mathcal{M}$ successfully generates the original objects for the test queries before applying the edit, and (b) $\mathcal{M}$ successfully generates o' when queried about e and r, namely, the edit has successfully been applied. For example, we verify that the model can predict the children of o' before asking about e's new siblings.

Editing Methods We evaluate three KE methods: MEND (Mitchell et al., 2022), ROME (Meng et al., 2022), and MEMIT (Meng et al., 2023). MEND trains a network that modifies gradients to produce local edits. ROME makes rank-one updates to the weights of the Transformer's MLP layers to modify specific factual associations, and

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

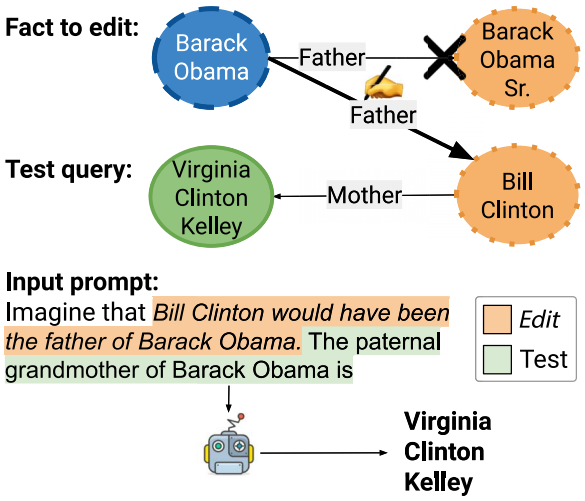


Figure 5: An example modification edit from our ICE baseline. The color code of the KG is similar to that described in Figure 2. We prepend the prefix ''*Imagine that*'' to the input prompt, as counterfactuals can contradict knowledge embedded in a model's parameters.

MEMIT is an extension of ROME that is adjusted to editing many facts at once.

Baseline Motivated by the recent success of LMs to learn in-context and follow instructions (Brown et al., 2020; Ouyang et al., 2022; Liu et al., 2023), specifically for knowledge editing (Zhong et al., 2023; Zheng et al., 2023), we experiment with an in-context editing (ICE) baseline for factual editing. Unlike the above methods, it does not introduce changes to the model parameters, but rather generation is conditioned on the new fact. Concretely, given an edit $(e, r, o) \rightarrow (e, r, o^*)$ and a test query q, we use the following prompt to obtain an answer from the model: ''`Imagine that <`o^*`> would have been <`P_r`>`'', where P_r is a manually-written proposition of r, such as ''*The mother of* <e>'' when $r =$ `Mother` and e is the subject. An example is depicted in Figure 5.

Models We use 4 recent auto-regressive decoder-only LMs of different sizes: GPT-2 XL (Radford et al., 2019) with 1.5B parameters, GPT-J (Chen et al., 2021) with 6B parameters, LLaMA with 7B parameters, (Touvron et al., 2023), and GPT-NeoX with 20B parameters (Black et al., 2022). In addition, as our baseline does not require access to the model parameters, we also evaluate it on the closed-source model GPT-3 `text-davinci-003` with 175B parameters (Brown et al., 2020). However, for the baseline we do not include results for GPT-2 and GPT-J as the number of testable edits for these models is rather small ($\leq 20\%$ for each of the data subsets).

	RECENT		RANDOM		POPULAR	
	Edits	Tests	Edits	Tests	Edits	Tests
GPT-2	853	29%	689	33%	722	71%
GPT-J	801	33%	717	34%	760	76%
GPT-NEO	989	45%	801	46%	828	86%
LLAMA	847	44%	796	49%	784	87%
GPT-3	822	55%	760	74%	665	94%

Table 2: (a) Number of edits considered in our evaluation (i.e., that have successfully applied), from each subset, averaged over ROME, MEMIT, and MEND, for the models: GPT-2, GPT-J, GPT-NEO and LLAMA, and the ICE baseline for GPT-3. (b) Portion of queries, on average, that were used in our evaluation.

For all model-method combinations, except for ROME with LLAMA, we use the official implementation and hyperparameters from Meng et al. (2022). We adjust ROME to LLAMA by following the authors' method and codebase. Table 2 shows the number of edits and test queries left, for every model, after filtering out non-successful edits and inapplicable test queries (as described above).

Evaluation Each model-method pair is evaluated separately, on every subset of RIPPLEEDITS. For each evaluation criterion, we first compute the average accuracy over the test queries per example, and then average over all the examples. For a given test query, we let the model generate a maximum of 20 tokens. A generation is considered successful if one of the aliases of the target object appears in the text. In cases of multiple gold target objects (as in *Preservation*), we evaluate each target object separately and consider the generation as correct if it matches at least one object.

5.2 Results

Tables 3, 4, 5 show the evaluation results on the RECENT, RANDOM, and POPULAR subsets, respectively. Considering the average scores across all subsets, we observe that existing editing methods struggle to handle the ripple effect induced by editing facts, with low average accuracy of 38–66 across all models. This suggests that, while

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

		LG	CI	CII	SA	PV	RS	Avg.
GPT-2	ROME	20.2	35.6	46.8	86.8	100	55.4	57.5
	MEMIT	21.8	30.3	46.2	92.9	100	56.8	58.0
	MEND	28.9	23.7	20.7	87.1	100	51.9	52.1
GPT-J	ROME	15.2	29.5	50.5	90.3	99.4	60.0	57.5
	MEMIT	18.0	35.0	48.1	88.4	98.6	42.2	55.0
GPT-NEO	ROME	27.2	54.3	69.4	98.9	98.4	80.3	71.4
	ICE	48.3	29.0	62.2	100	99.4	80.7	69.9
LLAMA	ROME	16.7	47.8	50.0	93.6	97.6	59.3	60.8
	ICE	59.6	74.8	85.0	100	99.5	77.9	82.8
GPT-3	ICE	33.3	100	91.3	100	100	73.1	82.8

Table 3: Accuracy on the RECENT subset, by MEND, ROME, MEMIT, and the ICE baseline, on GPT-2, GPT-J, GPT-NEO, LLAMA, and GPT-3.

		LG	CI	CII	SA	PV	RS	Avg.
GPT-2	ROME	53.6	31.6	44.4	94.9	9.9	38.9	45.5
	MEMIT	58.4	30.5	49.8	100	20.0	36.2	49.1
	MEND	62.5	16.7	14.6	91.3	17.7	30.1	38.8
GPT-J	ROME	53.8	40.8	49.9	93.8	15.2	39.4	48.8
	MEMIT	53.0	35.7	48.2	95.6	18.2	39.9	48.4
GPT-NEO	ROME	61.6	49.4	57.1	100	30.8	50.7	58.3
	ICE	78.6	90.0	55.6	100	100	61.9	81.0
LLAMA	ROME	54.3	35.5	49.5	96.0	17.8	38.9	48.7
	ICE	71.1	73.8	80.3	100	100	69.6	82.5
GPT-3	ICE	69.0	83.3	89.7	100	100	100	90.3

Table 4: Accuracy on the RANDOM subset, by MEND, ROME, MEMIT, and the ICE baseline, on GPT-2, GPT-J, GPT-NEO, LLAMA, and GPT-3.

KE methods demonstrate high capability in making local updates to the model's knowledge, these changes are mostly applied at a surface-level without propagating to other related facts. Moreover, we observe that our ICE baseline obtains the best overall results. Specifically, it outperforms ROME by more than 10 points for GPT-NEO and 29 points for LLAMA, on average across subsets. Although GPT-3 with ICE performs best on average, the 7B LLAMA is highly competitive, performing better or similarly on the RECENT and POPULAR subsets.

Next, comparing results across evaluation criteria shows that some ripple effects are handled better than others. For example, while *Subject Aliasing* accuracy is consistently high ($\geq$ 86.8 across all settings), the accuracy on other criteria is generally lower and varies greatly between models, methods, and edits (e.g., *Logical Generalization* accuracy for ROME on GPT-J is 53.8 on

		LG	CI	CII	SA	PV	RS	Avg.
GPT-2	ROME	5.7	46.4	21.8	100	100	18.5	48.7
	MEMIT	6.7	45.2	21.2	100	100	24.3	49.6
	MEND	25.9	10.7	5.4	100	100	21.2	43.9
GPT-J	ROME	5.5	44.1	21.0	98.6	99.0	22.3	48.4
	MEMIT	7.0	45.9	23.7	100	100	24.8	50.2
GPT-NEO	ROME	36.4	29.4	41.6	100	100	50.8	59.7
	ICE	37.5	92.4	40.1	100	100	74.4	74.1
LLAMA	ROME	22.0	37.4	16.2	100	100	20.6	49.4
	ICE	57.2	85.1	67.6	100	100	78.0	81.3
GPT-3	ICE	31.0	86.1	65.6	100	100	83.8	77.7

Table 5: Accuracy on the POPULAR subset, by MEND, ROME, MEMIT, and the ICE baseline, on GPT-2, GPT-J, GPT-NEO, LLAMA, and GPT-3.

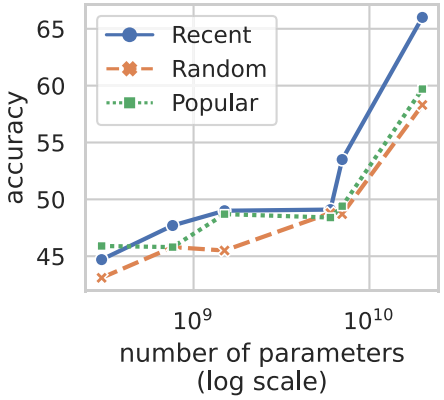


Figure 6: Accuracy averaged over evaluation criteria of ROME, as a function of the model's number of parameters, for the following models: GPT2-M, GPT2-L, GPT2-XL, GPT-J, LLAMA, and GPT-NEO.

the RANDOM subset, compared to only 5.5 on the POPULAR subset).

Results Across Model Size We analyze how editing performance on RIPPLEEDITS is influenced by the model size. To this end, we further evaluate ROME on smaller versions of GPT-2 – with 345M (GPT2-M) and 762M (GPT2-L) parameters, and plot the average accuracy over the three subsets as a function of model size. Figure 6 presents the results, showing that editing performance increases with model size, with ROME obtaining substantially higher accuracy when applied to larger models. Nevertheless, our results (Tables 3, 4, 5) show that when using ICE, the 7B LLAMA is competitive with the much larger GPT-3, suggesting that simply scaling the model size may not be sufficient to fix the drawbacks of current editing methods.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

	MEND	ROME	MEMIT
Relation Specificity	34.4	37.6	39.1
Logical Generalization	39.1	26.5	29.0
Compositionality I	17.0	37.9	35.3
Compositionality II	13.6	37.7	39.1

Table 6: Accuracy of MEND, ROME, and MEMIT, using GPT-2, averaged over the three RippleEdits splits - Recent, Random and Popular.

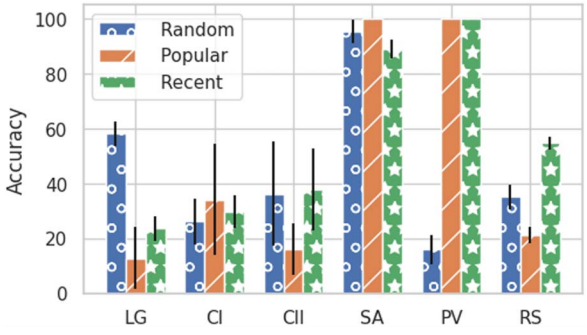


Figure 7: The average accuracy of GPT-2 on different evaluation criteria in RippleEdits. Results are averaged over editing methods (ROME, MEMIT, and MEND); error bars indicate standard deviation.

Results Across Methods Table 6 shows the accuracy of MEND, ROME, and MEMIT, on GPT-2 across our evaluation criteria, averaged over the three subsets. Interestingly, MEND outperforms ROME and MEMIT in *Logical Generalization*, but is worse in *Compositionality I* and *Compositionality II*, suggesting that different methods might better capture different types of ripple effects.

Results Across Data Splits The subsets of RippleEdits differ in whether edited facts are counterfeit or real, and in the popularity of the edited entities. These differences allow us to control for the edit severity, as popular entities are expected to introduce larger ripple effects (see §3). In Figure 7, we show the accuracy on each subset and evaluation criterion, averaged over the different editing methods. Comparing Random and Popular, that differ in the popularity of the edited entities, we see that while *Logical Generalization* accuracy is substantially higher for Random, *Preservation* accuracy is higher for Popular. This suggests that, although retaining correct knowledge is easier for popular entities, modifying other facts that logically follow from an edit is harder for popular entities, which could be explained by the severity of these edits (i.e., the high number of facts that are semantically related to them).

		No effect	Abstaining	Noise
GPT-2	ROME	27%	31%	42%
	ICE	32%	27%	41%
GPT-Neo	ROME	24%	40%	36%
	ICE	10%	65%	25%
LLaMA	ROME	20.5%	45%	34.5%
	ICE	11%	71%	18%

Table 7: Error type distribution on 200 failures of ROME and ICE, on GPT-2, GPT-Neo, and LLaMA.

5.3 Error Analysis

ROME versus ICE We qualitatively analyze the effect induced by KE methods to the model's knowledge. To this end, for each of ROME and our ICE baseline and each of the models GPT-2, GPT-Neo, and LLaMA, we sample 200 test queries from RippleEdits on which the model fails post-editing. We then label these failures using three categories: (a) *no effect*, for cases when the model predicts the original object, i.e., the edit introduced no ripple effect, (b) *abstaining*, when the model abstains from answering by generating text like *''unknown''* or *''a mystery''*, and (c) *noise*, when the model generates an incorrect object or unrelated text. Table 7 presents the results, showing that in most cases ($\geq 68\%$ across all settings) factual editing introduces erroneous changes to the model's knowledge rather than making no change. Interestingly, for both GPT-Neo and LLaMA, where editing performance is better than GPT-2, ROME introduces more incorrect changes while ICE causes the model to abstain from answering.

GPT-3 versus LLaMA using ICE We further looked into the performance on the LG tests, where applying ICE to GPT-3 is notably inferior to ICE on LLaMA (see Tables 3, 4, 5). Specifically, we collected responses from each of the models to 100 random LG queries, and analyzed them using the same categories as described above. We observed that GPT-3 abstains from answering the query much more often than LLaMA (49% of the cases for GPT-3 compared to only 28% in LLaMA), which could explain the lower performance of ICE on GPT-3 on these queries.

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

6 Conclusion and Discussion

We introduce the notion of ripple effects in knowledge editing, suggesting that editing a particular fact implies further updates of related facts. We additionally propose evaluation criteria for ripple effects and create RippleEdits, a diagnostic benchmark designed to evaluate how well KE methods handle the ripple effects of various edits. We evaluate prominent KE methods and show that they often fail to introduce consistent edits that capture the ripple effects of an edit, suggesting that future development of KE methods should consider those effects more carefully. Last, we show that a simple in-context editing method achieves the best results on RippleEdits, highlighting the potential of such editing approaches.

Notably, our benchmark covers a small fraction of all possible ripple-edits. For example, one could consider ripple effects that involve more than two hops, and explore the graph structure of different edits. In addition, while we focus on ripple effects of single edits, future work can consider the effect of editing multiple facts in a single batch. Finally, it would be interesting to consider cases where models succeed in capturing ripple-edits, and analyze how these are implemented mechanistically in the transformer architecture (Geva et al., 2023).

Limitations Our data generation pipeline relies on information from an existing knowledge-base (WikiData in our case), which could be incomplete or outdated. While RippleEdits does not aim to cover all the possible ripple-edits in WikiData, these concerns might be a major issue when seeking a comprehensive evaluation or considering domain-specific knowledge-bases, which often tend to be incomplete. A possible solution to explore in that case is to use LMs internal knowledge instead of an external knowledge-base (Cohen et al., 2023).

With RippleEdits focusing on the ripple effect of edits, it does not include tests, such as paraphrasing of the edit and subject specificity, that evaluate the edit itself and are covered by existing benchmarks (e.g., CounterFact). In addition, it does not verify that many other facts that are distantly related to the edit, i.e., triplets that are not included in the close neighbourhood of the edit, were retained post-editing. For example, we expect that editing the capital of France would not affect the population of Poland, yet this is not explicitly checked. We note that building such an evaluation is hard, since there are many facts to consider and it is unclear how to determine automatically which triplets should and should not be affected by a certain edit.

Acknowledgments

We thank Maor Ivgi and Gal Elidan for valuable feedback and constructive suggestions. This work is supported in part by the Israeli Science Foundation.

References

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. 2022. GPT-NeoX-20B: An open-source autoregressive language model. In *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.bigscience-1.9

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6–12, 2020, virtual*.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *ArXiv preprint*, abs/2107.03374. https://arxiv.org/abs/2107.03374

Roi Cohen, Mor Geva, Jonathan Berant, and Amir Globerson. 2023. Crawling the internal knowledge-base of language models. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1856–1869, Dubrovnik, Croatia. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-eacl.139

Damai Dai, Li Dong, Yaru Hao, Zhifang Sui, Baobao Chang, and Furu Wei. 2022. Knowledge neurons in pretrained transformers. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8493–8502, Dublin, Ireland. Association for Computational Linguistics. https://doi.org/10.18653/v1/2022.acl-long.581

Nicola De Cao, Wilker Aziz, and Ivan Titov. 2021. Editing factual knowledge in language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6491–6506, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.522

Bhuwan Dhingra, Jeremy R. Cole, Julian Martin Eisenschlos, Daniel Gillick, Jacob Eisenstein, and William W. Cohen. 2022. Time-aware language models as temporal knowledge bases. *Transactions of the Association for Computational Linguistics*, 10:257–273. https://doi.org/10.1162/tacl_a_00459

Konstantin Genin and Franz Huber. 2022. Formal representations of belief. In Edward N. Zalta and Uri Nodelman, editors, *The Stanford Encyclopedia of Philosophy*, Fall 2022 edition. Metaphysics Research Lab, Stanford University.

Mor Geva, Jasmijn Bastings, Katja Filippova, and Amir Globerson. 2023. Dissecting recall of factual associations in auto-regressive language models. *arXiv preprint arXiv:2304.14767*. https://doi.org/10.18653/v1/2023.emnlp-main.751

Mor Geva, Roei Schuster, Jonathan Berant, and Omer Levy. 2021. Transformer feed-forward layers are key-value memories. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5484–5495, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.446

Anshita Gupta, Debanjan Mondal, Akshay Krishna Sheshadri, Wenlong Zhao, Xiang Lorraine Li, Sarah Wiegreffe, and Niket Tandon. 2023. Editing commonsense knowledge in gpt. https://arxiv.org/abs/2305.14956

Peter Hase, Mona Diab, Asli Celikyilmaz, Xian Li, Zornitsa Kozareva, Veselin Stoyanov, Mohit Bansal, and Srinivasan Iyer. 2023. Methods for measuring, updating, and visualizing factual beliefs in language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2714–2731, Dubrovnik, Croatia. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.eacl-main.199

Benjamin Heinzerling and Kentaro Inui. 2021. Language models as knowledge bases: On entity representations, storage capacity, and paraphrased queries. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1772–1791, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.eacl-main.153

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023a. Inspecting and editing knowledge representations in language models. https://arxiv.org/abs/2304.00740

Evan Hernandez, Belinda Z. Li, and Jacob Andreas. 2023b. Measuring and manipulating knowledge representations in language models. *ArXiv preprint*, abs/2304.00740.

Jason Hoelscher-Obermaier, Julia Persson, Esben Kran, Ioannis Konstas, and Fazl Barez. 2023. Detecting edit failures in large language models: An improved specificity benchmark. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11548–11559, Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.findings-acl.733

Joel Jang, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. 2022. 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.

Saurav Kadavath, Tom Conerly, Amanda Askell, Tom Henighan, Dawn Drain, Ethan Perez, Nicholas Schiefer, Zac Hatfield Dodds, Nova DasSarma, Eli Tran-Johnson, Scott Johnston, Sheer El-Showk, Andy Jones, Nelson Elhage, Tristan Hume, Anna Chen, Yuntao Bai, Sam Bowman, Stanislav Fort, Deep Ganguli, Danny Hernandez, Josh Jacobson, Jackson Kernion, Shauna Kravec, Liane Lovitt, Kamal Ndousse, Catherine Olsson, Sam Ringer, Dario Amodei, Tom Brown, Jack Clark, Nicholas Joseph, Ben Mann, Sam McCandlish, Chris Olah, and Jared Kaplan. 2022. Language models (mostly) know what they know. *ArXiv preprint*, abs/2207.05221. https://arxiv.org/abs/2207.05221

Nora Kassner, Oyvind Tafjord, Hinrich Schütze, and Peter Clark. 2021. BeliefBank: Adding memory to a pre-trained language model for a systematic notion of belief. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8849–8861, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.697

Angeliki Lazaridou, Adhiguna Kuncoro, Elena Gribovskaya, Devang Agrawal, Adam Liska, Tayfun Terzi, Mai Gimenez, Cyprien de Masson d'Autume, Tomás Kociský, Sebastian Ruder, Dani Yogatama, Kris Cao, Susannah Young, and Phil Blunsom. 2021. Mind the gap: Assessing temporal generalization in neural language models. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 29348–29363.

Omer Levy, Minjoon Seo, Eunsol Choi, and Luke Zettlemoyer. 2017. Zero-shot relation extraction via reading comprehension. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 333–342, Vancouver, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/K17-1034

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35. https://doi.org/10.1145/3560815

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

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.546

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J. Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Eric Mitchell, Charles Lin, Antoine Bosselut, Chelsea Finn, and Christopher D. Manning. 2022. Fast model editing at scale. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yasumasa Onoe, Michael Zhang, Shankar Padmanabhan, Greg Durrett, and Eunsol Choi. 2023. Can LMs learn new entities from descriptions? Challenges in propagating injected knowledge. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5469–5485, Toronto, Canada. Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-long.300

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. https://arxiv.org/abs/2203.02155

Matthew E. Peters, Mark Neumann, Robert Logan, Roy Schwartz, Vidur Joshi, Sameer Singh, and Noah A. Smith. 2019. Knowledge enhanced contextual word representations. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 43–54, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1005

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics. https://doi.org/10.18653/v1/D19-1250

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Simon Razniewski, Andrew Yates, Nora Kassner, and Gerhard Weikum. 2021. Language models as or for knowledge bases. *ArXiv preprint*, abs/2110.04888. https://arxiv.org/abs/2110.04888

Adam Roberts, Colin Raffel, and Noam Shazeer. 2020. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.437

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV, Eric Wallace, and Sameer Singh. 2020. AutoPrompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4222–4235, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.emnlp-main.346

Chenglei Si, Zhe Gan, Zhengyuan Yang, Shuohang Wang, Jianfeng Wang, Jordan Lee Boyd-Graber, and Lijuan Wang. 2023. Prompting GPT-3 to be reliable. In *The Eleventh International Conference on Learning Representations*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux,

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026

Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models. *ArXiv preprint*, abs/2302.13971. https://arxiv.org/abs/2302.13971

Ruize Wang, Duyu Tang, Nan Duan, Zhongyu Wei, Xuanjing Huang, Jianshu Ji, Guihong Cao, Daxin Jiang, and Ming Zhou. 2021a. K-Adapter: Infusing knowledge into pre-trained models with adapters. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1405–1418, Online. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.findings-acl.121

Xiaozhi Wang, Tianyu Gao, Zhaocheng Zhu, Zhengyan Zhang, Zhiyuan Liu, Juanzi Li, and Jian Tang. 2021b. KEPLER: A unified model for knowledge embedding and pre-trained language representation. *Transactions of the Association for Computational Linguistics*, 9:176–194. https://doi.org/10.1162/tacl_a_00360

Yunzhi Yao, Shaohan Huang, Li Dong, Furu Wei, Huajun Chen, and Ningyu Zhang. 2022. Kformer: Knowledge injection in transformer feed-forward layers. In *Natural Language Processing and Chinese Computing*, pages 131–143, Cham. Springer International Publishing. https://doi.org/10.1007/978-3-031-17120-8_11

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. https://arxiv.org/abs/2305.13172

Ningyu Zhang, Shumin Deng, Xu Cheng, Xi Chen, Yichi Zhang, Wei Zhang, and Huajun Chen. 2021. Drop redundant, shrink irrelevant: Selective knowledge injection for language pretraining. In *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4007–4014. International Joint Conferences on Artificial Intelligence Organization. https://doi.org/10.24963/ijcai.2021/552

Xikun Zhang, Antoine Bosselut, Michihiro Yasunaga, Hongyu Ren, Percy Liang, Christopher D. Manning, and Jure Leskovec. 2022. Greaselm: Graph reasoning enhanced language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Zhengyan Zhang, Xu Han, Zhiyuan Liu, Xin Jiang, Maosong Sun, and Qun Liu. 2019. ERNIE: Enhanced language representation with informative entities. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1441–1451, Florence, Italy. Association for Computational Linguistics. https://doi.org/10.18653/v1/P19-1139

Ce Zheng, Lei Li, Qingxiu Dong, Yuxuan Fan, Zhiyong Wu, Jingjing Xu, and Baobao Chang. 2023. Can we edit factual knowledge by in-context learning? https://doi.org/10.18653/v1/2023.emnlp-main.296

Zexuan Zhong, Zhengxuan Wu, Christopher D. Manning, Christopher Potts, and Danqi Chen. 2023. Mquake: Assessing knowledge editing in language models via multi-hop questions. https://doi.org/10.18653/v1/2023.emnlp-main.971

Downloaded from http://direct.mit.edu/tacl/article-pdf/doi/10.1162/tacl_a_00644/2362212/tacl_a_00644.pdf by guest on 21 March 2026