Editing Factual Knowledge in Language Models

De Cao, N.; Aziz, W.; Titov, I.

DOI
10.18653/v1/2021.emnlp-main.522

Publication date
2021

Document Version
Final published version

Published in
2021 Conference on Empirical Methods in Natural Language Processing

License
CC BY

Link to publication

Citation for published version (APA):
De Cao, N., Aziz, W., & Titov, I. (2021). Editing Factual Knowledge in Language Models. In M.-C. Moens, X. Huang, L. Specia, & S. W. Sih (Eds.), *2021 Conference on Empirical Methods in Natural Language Processing: EMNLP 2021 : proceedings of the conference : November 7-11, 2021* (pp. 6491-6506). The Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.emnlp-main.522

Editing Factual Knowledge in Language Models

Nicola De Cao [1,2], Wilker Aziz [1], Ivan Titov [1,2]
[1]University of Amsterdam, [2]University of Edinburgh
{ nicola.decao, w.aziz, titov } @uva.nl

Abstract

The factual knowledge acquired during pre-training and stored in the parameters of Language Models (LMs) can be useful in downstream tasks (*e.g.*, question answering or textual inference). However, some facts can be incorrectly induced or become obsolete over time. We present KNOWLEDGEEDITOR, a method which can be used to *edit* this knowledge and, thus, fix 'bugs' or unexpected predictions without the need for expensive re-training or fine-tuning. Besides being computationally efficient, KNOWLEDGEEDITOR does not require any modifications in LM pre-training (*e.g.*, the use of meta-learning). In our approach, we train a hyper-network with constrained optimization to modify a fact without affecting the rest of the knowledge; the trained hyper-network is then used to predict the weight update at test time. We show KNOWLEDGEEDITOR's efficacy with two popular architectures and knowledge-intensive tasks: i) a BERT model fine-tuned for fact-checking, and ii) a sequence-to-sequence BART model for question answering. With our method, changing a prediction on the specific wording of a query tends to result in a consistent change in predictions also for its paraphrases. We show that this can be further encouraged by exploiting (*e.g.*, automatically-generated) paraphrases during training. Interestingly, our hyper-network can be regarded as a 'probe' revealing which components need to be changed to manipulate factual knowledge; our analysis shows that the updates tend to be concentrated on a small subset of components.[1]

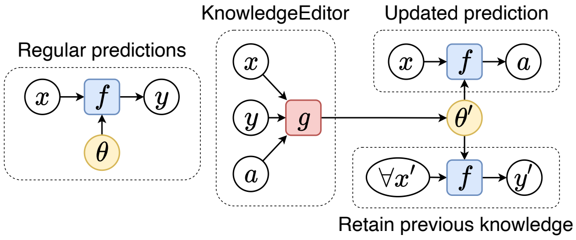


Figure 1: **Left:** a model f with parameters θ prefers a prediction y for input x (*e.g.*, y is the mode/argmax of a discrete distribution parameterized by $f(x;\theta)$). **Right:** our method uses a hyper-network g to update the parameters of f to θ' such that $f(x;\theta')$ prefers an alternative prediction a without affecting the prediction y' of *any* other input $x' \neq x$. Our model *edits the knowledge* about x stored in the parameters of f.

1 Introduction

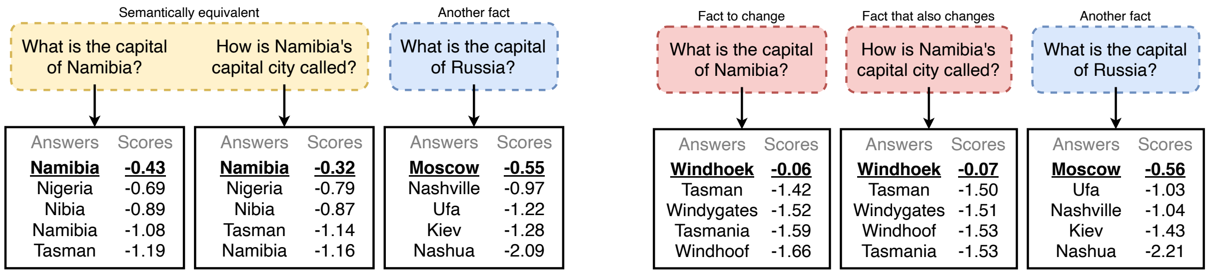
Using pre-trained transformer-based Language Models (LMs; Vaswani et al., 2017; Devlin et al., 2019; Radford et al., 2019; Lewis et al., 2020; Raffel et al., 2020; Brown et al., 2020) has recently become a standard practice in NLP. Factual knowledge induced during pre-training can help in downstream tasks, but it can also be incorrect or become obsolete over time (*e.g.*, not reflecting changes of heads of states or country populations). Developing reliable and computationally efficient methods for bug-fixing models without the need for expensive re-training would be beneficial. See Figure 2 for an example of revising the memory of a model that initially misremembered Namibia's capital.

Unlike conventional Knowledge Bases (KBs) that explicitly store factual knowledge, neural models implicitly memorize facts in their parameters. One cannot easily access and interpret their computation and memories (Ribeiro et al., 2016; Belinkov and Glass, 2019; Voita et al., 2019; De Cao et al., 2020), thus, modifying their knowledge is a challenging problem. Motivated by practical considerations, we formulate the following desiderata for a method aimed at tackling this problem (see Section 2 for a more formal treatment):

- **Generality:** be able to modify a model that was not specifically trained to be editable (*i.e.*, no need for special pre-training of LMs, such as using meta-learning);

[1]Source code available at https://github.com/nicola-decao/KnowledgeEditor

Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 6491–6506
November 7–11, 2021.



(a) Model predictions before the update.

(b) Model predictions with edited parameters.

Figure 2: Predictions from a pre-trained language BART model fine-tuned for closed-book question answering. **Left:** model top-k predictions from Beam Search. **Right:** top-k after using our method conditioning on changing '*What is the capital of Namibia?*' from '*Namibia*' (wrong) to '*Windhoek*' (correct prediction). Changing one fact also changes a semantically equivalent question and keeps the predictions from other facts the same.

- **Reliability:** be able to successfully update a specific fact without affecting the rest of the acquired knowledge;
- **Consistency:** the changes should be consistent across equivalent formulations of a fact (e.g., when asked to update an answer for one question, answers to its paraphrases should change accordingly).

The problem has been previously tackled in Zhu et al. (2020) and Sinitsin et al. (2020), as discussed in detail in Section 3. However, both do not ensure that the edited model will be 'reliable', *i.e.* that the rest of the knowledge would not be badly affected, and that the changes are 'consistent' across equivalent inputs. Additionally, Sinitsin et al.'s (2020) method requires expensive specialized training of the original network. While re-training the original network was feasible in their applications (e.g., in machine translation), it is problematic when the network is a pre-trained LM. We propose a novel method that overcomes these limitations.

We treat editing the memories of a neural model as a *learning-to-update* problem. We use an efficient parameterization of a hyper-network that is trained to update the LM parameters when provided with a single fact that needs to be modified. We do not require meta-learning, re-training or fine-tuning of the original network. We employ constrained optimization in training: we constrain the edited model to retain the same predictions as the original one regardless of the distance between the original and updated models in the parameter space. We show how this framework can be extended to incorporate (e.g., automatically-generated) paraphrases in training, further improving consistency. Figure 1 shows an outline of our method.

Differently from both previous methods, we do not have to select a subset of parameters to update as we let our model learn that by itself. In fact, our hyper-network can be regarded as a 'probe' revealing which components of the network need to be changed to manipulate factual knowledge, *i.e.* revealing the 'causal mediation mechanisms' (Vig et al., 2020). We observe that the updates end up being concentrated in a restricted set of model components, even though we do not encourage any kind of sparsity. Interestingly, the most-updated components are different from the groups of parameters receiving large gradients (see Figure 4).

Contributions Our contributions are as follows:

- we define the task of knowledge editing and propose a set of evaluation metrics;
- we propose KNOWLEDGEEDITOR that *learns* to modify LMs memories efficiently and reliably while maintaining consistent predictions for semantically equivalent inputs;
- we verify that our proposed method largely meets our desiderata—while other baselines based on fine-tuning fail—testing it with different LM architectures on knowledge-intensive tasks such as fact-checking and open-domain question answering;
- we analyze the updates for KNOWLEDGEEDITOR and the alternatives.

2 Task

We want to edit the memory of a neural language model such that when, presented with an input, its output reflects a revised collection of facts. Unfortunately, the knowledge of a language model is typically opaque to us, being stored non-locally across a large number of parameters and architectural components. Thus, concretely, to operationalize the task, we seek a change in the model's parameters that affects predictions from the model only for a specific input. For a given input x, the prediction a made by the edited model should differ from the prediction y made by the original model *only* if x is influenced by one of the revised facts.

2.1 Definition

More formally, we have a model $x \mapsto f(x;\theta)$ with trained parameters θ, and a dataset of revisions $\langle x, y, a\rangle \in \mathcal{D}$, *i.e.*, x is an input, y is the prediction preferred by $f(x;\theta)$, and a is an alternative prediction which we would like an edited version of the model to prefer. Concretely, we keep the model architecture f fixed, and seek alternative parameters θ' such that for x, $f(x;\theta')$ would prefer the prediction a instead of y while keeping all other predictions unchanged. In practice, we approximate the set of 'all other predictions' using a finite data set $\mathcal{O}^x$ of pairs $\langle x', y'\rangle$ with $x' \neq x$. Moreover, predictions need not be continuous nor differentiable outputs from the model; instead, they may result from an arbitrary decision rule based on $f(x;\theta)$. For example, when $f(x;\theta)$ parameterizes a discrete distribution $p_{Y|X}$ over the output space, the most standard decision rule is to output the mode of the distribution: $y = \arg\max_{c\in\mathcal{Y}}\ p_{Y|X}(c|x,\theta)$.[2]

Semantically equivalent inputs Optionally, for some revision $\langle x, y, a\rangle \in \mathcal{D}$, we may also have a set $\mathcal{P}^x$ of inputs semantically equivalent to x (*e.g.*, automatically-generated paraphrases). Such a set can be used in at least two ways: i) to obtain explicit supervision for changes that should be realized in tandem with $\langle x, y, a\rangle$; and, independently of that, ii) to evaluate whether an edited model makes consistent predictions on semantically equivalent inputs. Note that in this work we never use paraphrases at test time, only for training and evaluation of our approach; generating them at test time, while potentially helpful, would have compromised efficiency.

2.2 Evaluation

To test if a method g, producing edited parameters θ', meets our desiderata, we measure:

1. *success rate*: how much g successfully updates the knowledge in θ', measured as accuracy of revised predictions for inputs in $\mathcal{D}$;
2. *retain accuracy*: how well θ' retains the original predictions of f, measured as accuracy wrt input-output pairs in sets $\mathcal{O}^x$;
3. *equivalence accuracy*: how consistent the predictions of the revised model θ' are for semantically equivalent inputs, measured as accuracy of the revised predictions for all $\mathcal{P}^x$;
4. *performance deterioration*: how much test performance of the updated model deteriorates.[3]

These values are obtained by comparing predictions of $f(\cdot;\theta)$ and $f(\cdot;\theta')$ for different subsets of inputs (*e.g.*, $\mathcal{D}$, $\mathcal{O}^x$, $\mathcal{P}^x$) and against different targets (*e.g.*, gold-standard, original predictions, or alternative predictions). While these metrics are straightforward to compute in principle, some can be computationally demanding. For example, retain accuracy depends on predictions for *all* inputs we have access to, which is potentially the entirety of the downstream task's validation/test data.[4]

Previous work has evaluated similar versions of this task differently. Sinitsin et al. (2020) measure performance deterioration and success rate but do not measure *retain accuracy* nor *equivalence accuracy*. A small performance deterioration does not guarantee high equivalence accuracy as the former is sensitive to changes in cases where the original model makes wrong decisions. Assessing accuracy against old or revised facts, which Zhu et al. (2020) also do, does not help to measure the retain accuracy. We argue that preserving model predictions for inputs not in $\mathcal{D}$ is critical in production settings, where model predictions might have been extensively analyzed and tested. For $x' \notin \mathcal{D}$, we aim to maintain all original predictions as well as the model scores $f(x';\theta')$ itself, effectively avoiding the need to re-calibrate the models (for example, in applications where probability estimates are used downstream).

3 Related work

Modifying transformers The most straightforward strategy to edit the knowledge of a model would be to re-train it on a new dataset with additional, modified, or removed facts. This is often unfeasible as LMs require large-scale expensive training that can hardly be reproduced by the most.

[2] Whereas in text classification solving this is straightforward (for $\mathcal{Y}$ is small), in sequence-to-sequence we resort to beam search to approximate the mode (for $\mathcal{Y}$ is too large or unbounded).

[3] $1 - \frac{\text{accuracy of } f(\cdot;\theta')}{\text{accuracy of } f(\cdot;\theta)}$

[4] During training of g, however, we can use sub-sampling (*i.e.*, mini batches) to approximate the metric.

Sinitsin et al. (2020) propose a meta-learning approach (Finn et al., 2017) for model modification that learns parameters that are easily *editable* at test time (*e.g.*, updating the knowledge of the model requires only a few SGD steps from these learned parameters). To have a reliable method, they employ a regularized objective forcing the updated model not to deviate from the original one. This technique suffers from three main limitations: i) it requires expensive and specialized pre-training, ii) it is sensitive to many hyper-parameters (*e.g.*, the weights of the regularizers and the subset of parameters to update), and iii) their multitask objective does not *guarantee* reliability (*i.e.*, the model is penalized for diverging from the original, rather than constrained not to).

Instead of penalizing an updated model for deviating from the original one, Zhu et al. (2020) use constrained optimization. They use a less computationally expensive procedure as they *re*-fine-tune on a specific downstream task (with altered data). Their method employs either an L_2 or L_∞ constraint between the original model's parameters and the edited ones. However, a norm-based constraint on parameters ignores the highly non-linear nature of LMs and how parameters determine the outputs of the model. Indeed, a minimal change in parameter space may produce a completely different output for many datapoints leading to a potentially unreliable method. Additionally, they show the need to select a subset of parameters to be updated, which requires extra development effort. Zhu et al.'s (2020) method is similar to Elastic Weight Consolidation (Kirkpatrick et al., 2017), a technique developed for preventing catastrophic forgetting in neural network models.

Knowledge in Language Models Petroni et al. (2019) show that pre-trained language models recall factual knowledge without fine-tuning, which they do by feeding specific prompts to LMs. Handcrafted prompts have been found not to be the best option to extract knowledge from LMs, and various solutions have been proposed to understand what LMs 'know' (Jiang et al., 2020; Shin et al., 2020; Liu et al., 2021). Additionally, Roberts et al. (2020) show that large models can be fine-tuned to access their internal memories to answer questions in natural language without any additional context and with surprisingly high accuracy—a setting they referred to as closed-book question answering. Although performing quite well, these models cannot reach the prediction quality of alternatives that retrieve and use context. Approaches that incentivize memorization of factual knowledge show to be beneficial for many downstream tasks suggesting that research on methods that effectively edit the memory of a model is indeed important (Zhang et al., 2019; Sun et al., 2019, 2020). Some recent hybrid approaches that use both *implicit* and *explicit memory* show some benefits for question answering (Févry et al., 2020; Verga et al., 2020). Notably, language models that *only* rely on internal *implicit* memory are state-of-the-art for (multilingual-) Entity Linking (De Cao et al., 2021a,b). An effective mechanism for editing LM's implicit memory may be applicable in all these settings.

Causal Interventions Identification of minimal changes to neural networks needed to achieve a certain behaviour has been studied in the context of research in interpreting neural networks (Lakretz et al., 2019; Vig et al., 2020; Elazar et al., 2021; Csordás et al., 2021). The components which need to be updated can be interpreted as controlling or encoding the corresponding phenomena (e.g., subject-verb agreement). Much of this research focused on modifying neuron activations rather than weights and on sparse interventions (e.g., modifying one or a handful of neurons). While far from our goals, there are interesting connections with our work. For example, our analysis of updates in Section 6.4, though very limited, may shed some light on how factual knowledge is encoded in the parameters of a model.

4 Method

We propose to treat the task of editing the memory of a neural model as a learning problem. Instead of defining a handcrafted algorithm to compute the new parameters θ', we learn a KNOWLEDGEEDITOR: a model that predicts θ' conditioned on an atomic fact that we want to modify. Concretely, KNOWLEDGEEDITOR is a hyper-network (Ha et al., 2017)—*i.e.*, a neural network that predicts the parameters of another network. Since the task requires every other prediction to stay the same—except the one we desire to change—we cast the learning task as a constrained optimization problem.

Optimization For an input x, changing the prediction of a model $f(\cdot;\theta)$ to a corresponds to minimizing the loss $\mathcal{L}(\theta;x,a)$ incurred when a is the target. Preserving the rest of the knowledge corresponds to constraining the updated parameter θ' such that model outputs $f(\cdot;\theta')$ do not change for $x' \in \mathcal{O}^x$. Our editor g is a neural network parameterized by ϕ which we choose by optimising the following objective for each data-point $\langle x, y, a \rangle \in \mathcal{D}$:

$$\begin{aligned} \min_{\phi} \quad & \sum_{\hat{x} \in \mathcal{P}^x} \mathcal{L}(\theta'; \hat{x}, a) \\ \text{s.t.} \quad & \mathcal{C}(\theta, \theta', f; \mathcal{O}^x) \leq m \ , \end{aligned} \tag{1}$$

where $\mathcal{P}^x$ is the set of semantically equivalent inputs to x (for convenience we assume it contains at least x), $\theta' = \theta + g(x, y, a; \phi)$, $\mathcal{C}$ is a constraint on the update, and the margin $m \in \mathbb{R}_{>0}$ is a hyperparameter. The constraint is used to express our desire to preserve model outputs unchanged for $x' \neq x$. Note that only x, but not the rest of $\mathcal{P}^x$, are provided as input to the editor, as these will not be available at test time. In our models, $f(x;\theta)$ parameterizes a discrete distribution $p_{Y|X}$ over the output sample space $\mathcal{Y}$, hence we choose to constrain updates in terms of sums of Kullback-Leibler (KL) divergences from the updated model to the original one: $\mathcal{C}_{KL}(\theta, \theta', f; \mathcal{O}^x) =$

$$\sum_{x' \in \mathcal{O}^x} \sum_{c \in \mathcal{Y}} p_{Y|X}(c|x', \theta) \log \frac{p_{Y|X}(c|x', \theta)}{p_{Y|X}(c|x', \theta')} \tag{2}$$

The constraint pushes the updated model to predict output distributions identical to the original one for all $x' \neq x$. An alternative constraint we could employ is an L_p norm over the parameter updates such that g is optimized to make a minimal update to the original model parameter: $\mathcal{C}_{L_p}(\theta, \theta', f; \mathcal{O}^x) = (\sum_i |\theta_i - \theta'_i|^p)^{1/p}$. This constraint was previously used by Zhu et al. (2020). However, such a constraint, expressed purely in parameter space and without regards to the model architecture f, does not directly encourage model outputs to be close to original ones in *function space* (*i.e.*, the two functions to be similar). Neural models are highly non-linear functions, so we do not expect this type of constraint to be effective. This will be empirically demonstrated in Section 6.

Tractable approximations Non-linear constrained optimization is generally intractable, thus we employ Lagrangian relaxation (Boyd et al., 2004) instead. The constraint itself poses a computational challenge, as it requires assessing KL for all datapoints in the dataset at each training step. For tractability, we evaluate the constraint approximately via Monte Carlo (MC) sampling (see Appendix A for more details). Finally, in sequence-to-sequence models, assessing KL is intractable even for a single data point, as the sample space $\mathcal{Y}$ is unbounded. In such cases we approximate the computation on a subset of the sample space obtained via beam search.

Architecture Instead of predicting θ' directly, our hyper-network predicts a shift $\Delta\theta$ such that $\theta' = \theta + \Delta\theta$. A *naive* hyper-network implementation might be over-parameterized, as it requires a quadratic number of parameters with respect to the size of the target network. Thus, we apply a trick similar to Krueger et al. (2017) to make g tractably predict edits for modern large deep neural networks (*e.g.*, BERT). Namely, g makes use of the gradient information $\nabla_\theta \mathcal{L}(\theta; x, a)$ as it carries rich information about how f accesses the knowledge stored in θ (*i.e.*, which parameters to update to increase the model likelihood given a).[5]

We first encode $\langle x, y, a \rangle$, concatenating the text with special separator and feeding it to a bidirectional-LSTM (Hochreiter and Schmidhuber, 1997). Then, we feed the last LSTM hidden states to a FFNN that outputs a single vector h that conditions the further computations. To predict the shift for a weight matrix $W^{n \times m} \in \theta$, we use five FFNNs conditioned on h that predict vectors $\alpha, \beta \in \mathbb{R}^m, \gamma, \delta \in \mathbb{R}^n$ and a scalar $\eta \in \mathbb{R}$. Then

$$\begin{aligned} \Delta W &= \sigma(\eta) \cdot \left(\hat{\alpha} \odot \nabla_W \mathcal{L}(W; x, a) + \hat{\beta} \right) \ , \\ \text{with} \quad & \hat{\alpha} = \hat{\sigma}(\alpha) \gamma^\top \quad \text{and} \quad \hat{\beta} = \hat{\sigma}(\beta) \delta^\top \ , \end{aligned} \tag{3}$$

where σ is the Sigmoid function (*i.e.*, $x \mapsto (1 + \exp(-x))^{-1}$), and $\hat{\sigma}$ indicates the Softmax function (*i.e.*, $x \mapsto \exp(x)/\sum_i \exp(x_i)$). With this formulation, the parameters for the hyper-network ϕ scale linearly with the size of θ. An interpretation of Equation 3 is that an update ΔW is a gated sum of a scaled gradient of the objective and a bias term. The scale for the gradient and the bias are generated via an outer vector product as it allows for efficient parameterization of a matrix with just three vectors. The gate lets the model keep some parameters unchanged.

Margin annealing The margin m is a hyperparameter and therefore fixed. However, i) it is hard to choose since it is task-dependent, and ii) it should

[5] A version of our hyper-network that does not use gradient information converges far too slowly.

be as small as possible. If the margin is too small, however, we risk having a small feasible set, and the model may never converge. To address both issues, we pick some initial value for the margin and anneal it during training conditioned on validation performance: when the model successfully changes $> 90\%$ of the predictions, we multiply the margin by 0.8. We stop decreasing the margin once it reaches a desirable small value. The annealing procedure prevents the model from diverging while increasingly tightening the constraint.

5 Experimental Setting

We aim to evaluate the effectiveness of KNOWLEDGEEDITOR comparing to baselines on knowledge-intensive tasks where the importance of modifying the memory of a large LM has a broad impact. We then test our method on closed-book fact-checking and closed-book question answering with the metrics proposed in Section 2.2.

5.1 Baselines

We compare against two baselines: i) fine-tuning and ii) the method proposed by Zhu et al. (2020). Fine-tuning corresponds to using standard gradient descent, minimizing the loss for the fact/prediction we want to revise. For this, we follow Sinitsin et al. (2020) and employ RMSProp (Tieleman and Hinton, 2012).[6] We set the learning rate to 10^{-5} and stop upon successfully changing the output of the model or having reached a maximum of 100 gradient steps. Zhu et al.'s (2020) method extends fine-tuning with an L_∞ constraint on parameters.[7] Following both Sinitsin et al. (2020) and Zhu et al. (2020) we report these baselines fine-tuning all parameters or just a subset of them. We limit the search to selecting entire layers and base our decision on performance on a subset of the validation set. Note that selecting a subset of parameters for update requires an extensive search, which KNOWLEDGEEDITOR dispenses with by automatically learning it.

5.2 Models and data

We evaluate on closed-book fact-checking (FC) fine-tune a BERT base model (Devlin et al., 2019) on the binary FEVER dataset (Thorne et al., 2018) from KILT (Petroni et al., 2021). We also evaluate on a task with a more complex output space: closed-book question answering (QA). For that we fine-tune a BART base model (Lewis et al., 2020) with a standard seq2seq objective on the Zero-Shot Relation Extraction (zsRE) dataset by Levy et al. (2017). We evaluate on this dataset because it is annotated with human-generated question paraphrases that we can use to measure our model's robustness to semantically equivalent inputs. We create alternative predictions for FC simply flipping the labels, whereas for QA we pick all hypotheses enumerated via beam search except the top-1. The latter ensures high-probability outcomes under the model distribution. We generate semantically equivalent inputs with back-translation. See Appendix B for technical details on models and data collection.

[6] We tried alternatives, RMSProp was the most effective.

[7] We search the hyper-parameter for the penalty $m \in \{10^{-3}, 5\times10^{-4}, 10^{-4}, 5\times10^{-5}, 10^{-5}\}$ selecting the best based on the sum of success rate and retain accuracy.

6 Results

Table 1 reports the main results for fact-checking and question answering. Overall, KNOWLEDGEEDITOR achieves high performance in all metrics. Some other methods also achieve high accuracy in some metrics but always sacrificing others (*i.e.*, never meeting all our desiderata at once).

We compare methods along different metrics (as opposed to a single one), as there is no way to precisely determine the importance of each of these metrics. To gather more insight, we compute their stochastic convex combination with coefficients sampled from a Dirichlet distribution (with $\alpha = 1$ to ensure a very diverse set of combinations) and report in Figure 6 in Appendix C an estimate of the probability that a system outperforms another across $1,000$ such combinations. The probability of our full method to outperform all baselines is very high for both FC and QA ($\approx$97% and $\approx$88%, respectively). In Figure 5 in Appendix C, we show the distributions of the combined scores (*i.e.*, the raw data for the approximation reported in Figure 6). We then analyze different aspects of our method and the baselines.

6.1 Success rate

Every method achieves an almost perfect success rate on fact-checking. All methods but ours apply updates in a loop, stopping either when the new model is successfully updated or after reaching a maximum number of iterations. The success rate for KNOWLEDGEEDITOR is not 100% because we do not apply more than one update even in case of failure. To this end, we also show an experiment

Method	Fact-Checking Success rate ↑	Retain acc ↑	Equiv. acc ↑	Perform. det ↓	Question Answering Success rate ↑	Retain acc ↑	Equiv. acc ↑*	Perform. det ↓
Fine-tune (1st layer)	100.0	99.44	42.24	0.00	98.68	91.43	89.86 / 93.59	0.41
Fine-tune (all layers)	100.0	86.95	95.58	2.25	100.0	67.55	97.77 / 98.84	4.50
Zhu et al. (1st layer)	100.0	99.44	40.30	0.00	81.44	92.86	72.63 / 78.21	0.32
Zhu et al. (all layers)	100.0	94.07	83.30	0.10	80.65	95.56	76.41 / 79.38	0.35
Ours $\mathcal{C}_{L_2}$	99.10	45.10	99.01	35.29	99.10	46.66	97.16 / 99.24	9.22
KnowledgeEditor	98.80	98.14	82.69	0.10	94.65	98.73	86.50 / 92.06	0.11
+ loop†	100.0	97.78	81.57	0.59	99.23	97.79	89.51 / 96.81	0.50
+ $\mathcal{P}^x$ ‡	98.50	98.55	95.25	0.24	94.12	98.56	91.20 / 94.53	0.17
+ $\mathcal{P}^x$ + loop‡	100.0	98.46	94.65	0.47	99.55	97.68	93.46 / 97.10	0.95

Table 1: Accuracy scores on fact-checking and question answering for the metrics presented in Section 2.2. *We report both the accuracy on the set of generated paraphrases (left) and human-annotated (right).†Apply updates in a loop, stopping when the update is a success or when reaching a maximum number of iterations (only at test time). ‡Using paraphrases (semantically equivalent inputs) as additional supervision (only at training time).

with our method with multiple updates within a *loop* employing the same stopping criteria as the baselines. Note that we apply this only at test time (*i.e.*, we do not train for multiple updates). When applying multiple updates also our method reaches a 100% success rate on fact-checking and almost perfect accuracy ($> 99\%$) for QA.[8]

Closed-book QA is a more challenging task since the output space is text and not just a binary label. In this setting, KnowledgeEditor achieves high accuracy ($\approx 95\%$ or $> 99\%$ with the *loop*). Among all methods, KnowledgeEditor gets the best success rate while also obtaining the best retain accuracy. In QA, Zhu et al.'s (2020) method does not reach a good success rate ($\approx 80\%$). We searched hyperparameters for their method also to have high retain accuracy, and indeed that is higher than regular fine-tuning. However, unlike fact-checking, regular fine-tuning for QA gets almost perfect scores but at the expense of the retain accuracy. Sequence-to-sequence models are more sensitive to a slight parameter shift. This happens because minor changes may completely alter the top-k prediction from beam search (in the case of QA). Differently, in a binary classifier (in the case of FC) the probability of a prediction can change substantially without crossing the decision boundary (usually set at 0.5 when not calibrated).

6.2 Retaining previous knowledge

KnowledgeEditor maintains the predictions in the validation set almost perfectly (retain accuracy is $\approx 98\%$ for both FC and QA). Conversely, as expected, our method with $\mathcal{C}_{L_2}$ has very low retain accuracy (always $< 50\%$). $\mathcal{C}_{L_2}$ suffers from catastrophic forgetting because it does not enforce the updated model to be close to the original one in function space (*i.e.*, the two functions to be similar) but just in parameter space.

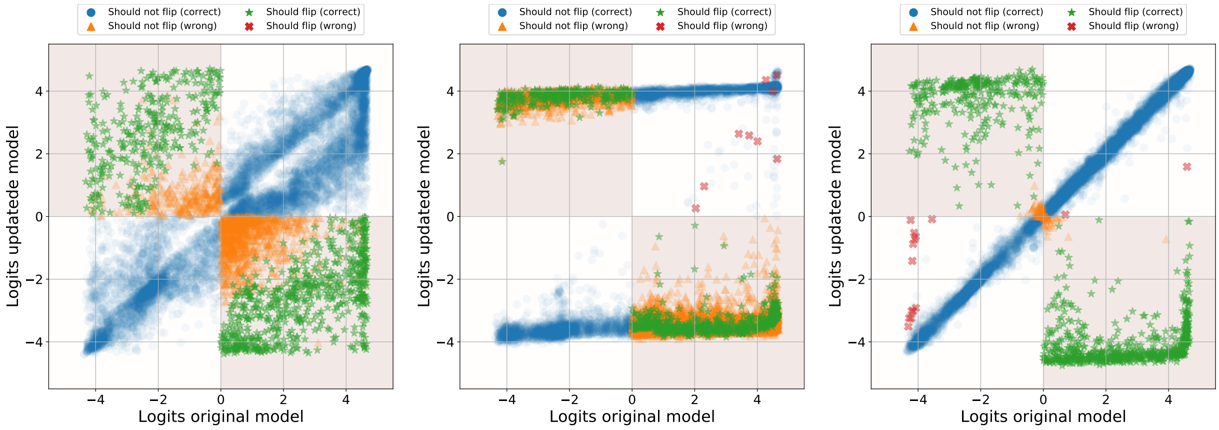
Fine-tuning all layers is successful but it affects the previously acquired knowledge negatively: retain accuracy is $\approx 87\%$ and $\approx 68\%$ for FC and QA, respectively, while performance deterioration in $\approx 2\%$ and $\approx 4\%$. Fine-tuning a single layer is more effective as it prevents over-fitting (the best model updates the 1st layer in both FC and QA). However, in FC the updated model does not generalize on semantic equivalent inputs: the accuracy on paraphrases is much lower even than versions of our methods which do not use paraphrases in training (42% vs. $> 81\%$), and even more so when compared to those which use them ($> 94\%$).

Fine-tuning with Zhu et al.'s (2020) method does not affect performance for FC much, which is not surprising since standard fine-tuning already gets almost perfect scores. Differently, in the QA setting, using their constrained optimization boosts the retain accuracy (up to $+4\%$ to normal fine-tuning) but at the cost of a low success rate ($\approx 80\%$ where fine-tuning gets the perfect score).

6.3 Accuracy on paraphrases

We evaluate our method both with and without the additional supervision of paraphrases to improve generalization—that corresponds to have $\mathcal{P}^x$ as the set of paraphrases of x or $\mathcal{P}^x = \{x\}$ in Equation 1, respectively. Without this additional supervision,

[8]Even if we do not train for multiple subsequent updates, its success opens the possibility to add this at training time. We leave the exploration of this technique to future work.



(a) Fine-tune (all layers).   (b) $\mathcal{C}_{L_2}$.   (c) Ours $\mathcal{C}_{KL}$ with $\mathcal{P}^x$.

Figure 3: Distribution of logits of the original model and updated model on FEVER. Fine-tuning all layers (a) leads to many errors, and the probability of the predictions does not stay the same even when they do not cross the decision boundary. $\mathcal{C}_{L_2}$ (b) successfully flips labels, but it does not force the predictions to stay the same. For our full method, $\mathcal{C}_{KL}$ with $\mathcal{P}^x$ (c), errors are mainly concentrated around the origin where the model is uncertain, and small perturbations make logits to cross the decision boundary. *Better view with colors.*

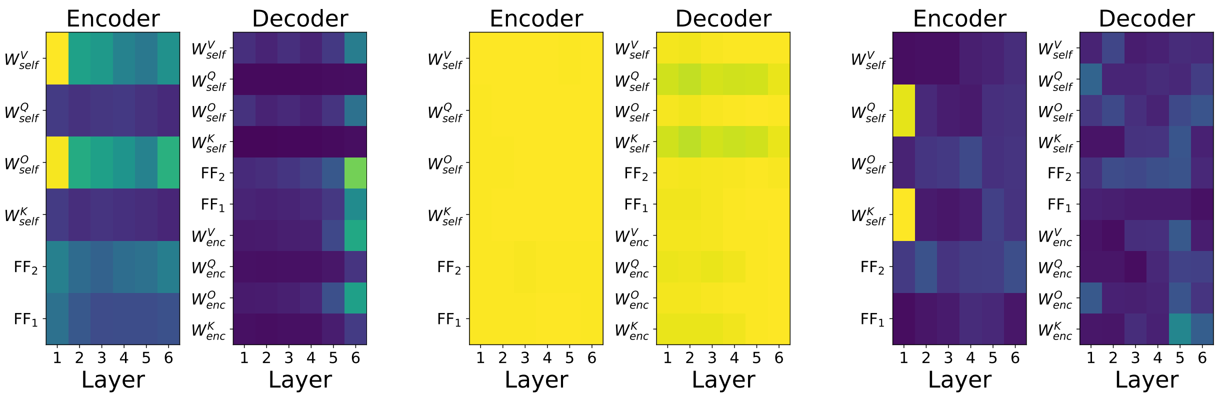
KNOWLEDGEEDITOR is already competitive in equivalence accuracy. However, employing this additional supervision is clearly beneficial on both tasks: we get the same success rate and re-train accuracy but equivalence accuracy improves by $> 70\%$ on FC and $> 30\%$ on QA, respectively (for generated paraphrases). In FC, although fine-tuning of a single layer proved to be optimal in terms of success rate and retain accuracy, it performs poorly for paraphrases. That is the model successfully updates the prediction of a particular datapoint, but does not update predictions of paraphrases. This indicates that fine-tuning to edit the knowledge of a model does not generalize well, and it *overfits* to specific inputs. On QA, also Zhu et al. (2020) performs poorly compared to our or other methods.

When other methods perform on par or better than ours on paraphrases, they do not have good retain accuracy (*e.g.*, see QA fine-tuning on Table 1). Fine-tuning on QA seems to generalize better than on FC, but does not preserve previous knowledge. In Table 1 we also report both the accuracy on the set of generated and human-generated paraphrases. Surprisingly, the scores on human-generated paraphrases are higher. We speculate that this happens because automatic paraphrases are sometimes not semantically equivalent or fluent.

6.4 Analysis of model updates

In Figure 3 we plot the distribution of logits of the original and updated model on FC for different methods. With an ideal method, all logits before and after an update have to stay the same (except the ones we want to change). From that figure, we can see distributions of different types of errors such as datapoints whose predictions were mistakenly flipped (from true to false or the other way around). These errors are mostly concentrated around the origin, where small perturbations make logits cross the decision boundary. When fine-tuning all layers, we can see a clear impact on logits, they undergo a lot of change (*i.e.*, points do not concentrate around the diagonal). Indeed, fine-tuning makes many datapoints cross the decision boundary and their probabilities to change from the original ones. The failure of $\mathcal{C}_{L_2}$ is visible in Figure 3b as this method preserves almost none of the previous predictions. Instead KNOWLEDGEEDITOR preserves almost all of the predicted labels as well as their probabilities (most datapoints in Figure 3c stay on the diagonal).

We also report visualizations of the average weight updates for the QA experiment in Figure 4. We report the setting with additional supervision from paraphrases (but the heatmaps are similar without them). There are three main observations from this plot. First, gradients are mostly concentrated on the first encoder layer and the last decoder layer. Gradients explain why the best subset of parameters to update is the first layer. Secondly, fine-tuning does not preserve gradient magnitudes and updates the whole model almost uniformly. That happens because of the optimizer's adaptive



(a) Gradients. (b) Fine-tune (all layers). (c) KNOWLEDGEEDITOR + $\mathcal{P}^x$.

Figure 4: Average normalized magnitude of updates on weight matrices across layers for the QA experiment. Fine-tuning updates all layers uniformly while our updates are more sparse.

learning rate that initially erases the gradient direction. The gradient direction plays a role only after a couple of gradient steps, but most of the time, the method only needs one step to modify its knowledge. Lastly, our updates are sparser and are not consistent with the gradient for changing the predictions. That indicates that our method learns to use the gradient in a meaningful way (*i.e.* ignoring some directions or manipulating its magnitude). It is surprising that the knowledge manipulation seems to be achieved by primarily modifying parameters affecting the shape of the attention distribution (W^K_{self} and W^Q_{self}) rather than, *e.g.*, values (W^V_{self}). As we discussed, the hyper-network may be regarded as a probe providing insights about the mechanism used by the model to encode the knowledge (Vig et al., 2020). For example, the focus on the bottom layer is already intriguing, as it contrasts with claims that memorization happens in top layers of image classification models (Stephenson et al., 2021), hinting at substantial differences in the underlying memorization mechanisms in NLP and vision. Proper investigation is however outside of the scope of this study. See Appendix C for some additional analysis.

7 Conclusions

In this work, we explore the task of editing the factual knowledge implicitly stored in the parameters of Language Models. For this task, we formally define desiderata, the objective, and a set of metrics to measure the efficacy of different methods. We concretely evaluate that on two benchmarks based on closed-book fact-checking and question answering. We propose KNOWLEDGEEDITOR, a method based on a hyper-network that *learns* to modify implicit knowledge stored within LM parameters efficiently and reliably. We provide comprehensive evaluations for our models against different variants of fine-tuning demonstrating the advantage of our approach. The magnitude of the updates predicted by our method may unfold the mechanisms used by the LMs to encode factual knowledge; we leave such investigation for future work.

Ethical Considerations

Technology built upon pre-trained LMs inherits some or all of their potential harms (Bender et al., 2021). Our technology for editing the knowledge of LMs does not exacerbate their potential harms and can, in fact, be used to mitigate harms, as models can be corrected once problems are discovered. However, we note that malicious uses of our knowledge editor are possible. For example, malicious agents may use the techniques presented in this work to inject incorrect knowledge into LMs.

Acknowledgments

The authors want to thank Michael Schlichtkrull, Lena Voita and Luisa Quarta for helpful discussions and support. This project is supported by SAP Innovation Center Network, ERC Starting Grant BroadSem (678254), the Dutch Organization for Scientific Research (NWO) VIDI 639.022.518, and the European Union's Horizon 2020 research and innovation programme under grant agreement No 825299 (Gourmet).

References

Yonatan Belinkov and James Glass. 2019. Analysis methods in neural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 7:49–72.

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Stephen Boyd, Stephen P Boyd, and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

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*.

Róbert Csordás, Sjoerd van Steenkiste, and Jürgen Schmidhuber. 2021. Are neural nets modular? inspecting functional modularity through differentiable weight masks. In *Submitted to International Conference on Learning Representations*.

Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2021a. Autoregressive entity retrieval. In *International Conference on Learning Representations*.

Nicola De Cao, Michael Sejr Schlichtkrull, Wilker Aziz, and Ivan Titov. 2020. How do decisions emerge across layers in neural models? interpretation with differentiable masking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3243–3255, Online. Association for Computational Linguistics.

Nicola De Cao, Ledell Wu, Kashyap Popat, Mikel Artetxe, Naman Goyal, Mikhail Plekhanov, Luke Zettlemoyer, Nicola Cancedda, Sebastian Riedel, and Fabio Petroni. 2021b. Multilingual autoregressive entity linking. *arXiv preprint arXiv:2103.12528*.

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

Yanai Elazar, Shauli Ravfogel, Alon Jacovi, and Yoav Goldberg. 2021. Amnesic probing: Behavioral explanation with amnesic counterfactuals. *Transactions of the Association for Computational Linguistics*, 9:160–175.

Thibault Févry, Livio Baldini Soares, Nicholas FitzGerald, Eunsol Choi, and Tom Kwiatkowski. 2020. Entities as experts: Sparse memory access with entity supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4937–4951, Online. Association for Computational Linguistics.

Chelsea Finn, Pieter Abbeel, and Sergey Levine. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning, ICML 2017, Sydney, NSW, Australia, 6-11 August 2017*, volume 70 of *Proceedings of Machine Learning Research*, pages 1126–1135. PMLR.

David Ha, Andrew M. Dai, and Quoc V. Le. 2017. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. 2018. Marian: Fast neural machine translation in C++. In *Proceedings of ACL 2018, System Demonstrations*, pages 116–121, Melbourne, Australia. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations,* *ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

J. Kirkpatrick, Razvan Pascanu, Neil C. Rabinowitz, J. Veness, G. Desjardins, Andrei A. Rusu, K. Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, C. Clopath, D. Kumaran, and Raia Hadsell. 2017. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114:3521 – 3526.

David Krueger, Chin-Wei Huang, Riashat Islam, Ryan Turner, Alexandre Lacoste, and Aaron Courville. 2017. Bayesian hypernetworks. *arXiv preprint arXiv:1710.04759*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:453–466.

Yair Lakretz, German Kruszewski, Theo Desbordes, Dieuwke Hupkes, Stanislas Dehaene, and Marco Baroni. 2019. The emergence of number and syntax units in LSTM language models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 11–20, Minneapolis, Minnesota. Association for Computational Linguistics.

Nayeon Lee, Belinda Z. Li, Sinong Wang, Wen-tau Yih, Hao Ma, and Madian Khabsa. 2020. Language models as fact checkers? In *Proceedings of the Third Workshop on Fact Extraction and VERification (FEVER)*, pages 36–41, Online. Association for Computational Linguistics.

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.

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

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. GPT Understands, Too. *arXiv preprint arXiv:2103.10385*.

Fabio Petroni, Aleksandra Piktus, Angela Fan, Patrick Lewis, Majid Yazdani, Nicola De Cao, James Thorne, Yacine Jernite, Vladimir Karpukhin, Jean Maillard, Vassilis Plachouras, Tim Rocktäschel, and Sebastian Riedel. 2021. KILT: a benchmark for knowledge intensive language tasks. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2523–2544, Online. Association for Computational Linguistics.

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.

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.

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

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. 2016. Model-agnostic interpretability of machine learning. *International Conference on Machine Learning (ICML) Workshop on Human Interpretability in Machine Learning*.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

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.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958.

Cory Stephenson, Suchismita Padhy, Abhinav Ganesh, Yue Hui, Hanlin Tang, and SueYeon Chung. 2021. On the geometry of generalization and memorization in deep neural networks. *Proceedings of International Conference on Learning Representations (ICLR)*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Hao Tian, Hua Wu, and Haifeng Wang. 2020. Ernie 2.0: A continual pre-training framework for language understanding. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8968–8975.

Ilya Sutskever, James Martens, and Geoffrey E. Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 1017–1024. Omnipress.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jonathon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 2818–2826. IEEE Computer Society.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 809–819, New Orleans, Louisiana. Association for Computational Linguistics.

Tijmen Tieleman and Geoffrey Hinton. 2012. Lecture 6.5—RmsProp: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2):26–31.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008.

Pat Verga, Haitian Sun, Livio Baldini Soares, and William W Cohen. 2020. Facts as experts: Adaptable and interpretable neural memory over symbolic knowledge. *arXiv preprint arXiv:2007.00849*.

Jesse Vig, Sebastian Gehrmann, Yonatan Belinkov, Sharon Qian, Daniel Nevo, Yaron Singer, and Stuart Shieber. 2020. Causal mediation analysis for interpreting neural NLP: The case of gender bias. *NeurIPS*.

Elena Voita, Rico Sennrich, and Ivan Titov. 2019. The bottom-up evolution of representations in the transformer: A study with machine translation and language modeling objectives. 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 4396–4406, Hong Kong, China. Association for Computational Linguistics.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

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.

Chen Zhu, Ankit Singh Rawat, Manzil Zaheer, Srinadh Bhojanapalli, Daliang Li, Felix Yu, and Sanjiv Kumar. 2020. Modifying memories in transformer models. *arXiv preprint arXiv:2012.00363*.

A Relaxation and Approximation of Constrained Optimization

Given a objective to minimize in the form of

$$\begin{aligned} \min_{\phi} \quad & \mathop{\mathbb{E}}_{x \sim p(x)} [f(x, \theta)] \\ \text{s.t.} \quad & \frac{1}{|\mathcal{Y}|} \sum_{x \in \mathcal{Y}} \mathcal{C}(y, \theta) \leq m \,, \end{aligned} \tag{4}$$

can be solved with Lagrangian relaxation (Boyd et al., 2004) using a multiplier $\alpha \in \mathbb{R}_{\geq 0}$ and be approximated by sampling $y \sim p(y)$ to

$$\min_{\phi} \max_{\alpha} \; f(x, \theta) + \alpha \cdot (\mathcal{C}(y, \theta) - m) \;. \tag{5}$$

Equation 5 can be evaluated with automatic differentiation and optimized via gradient descent.

B Experimental setting

B.1 Fact-checking

We evaluate on closed-book fact-checking (FC) using the binary FEVER dataset (Thorne et al., 2018) from KILT (Petroni et al., 2021). FEVER has 104,966 training and 10,444 validation instances respectively. For every input claim x, the model predicts the probability $f(x; \theta)$ that it may be true. This is done *without* retrieving any evidence from a corpus, instead, just by relying on the knowledge accumulated during pre-training and encoded in its own parameters—this is similar to Lee et al. (2020) that investigate closed-book and zero-shot FC using masked-LMs. Concretely, we ask the LM to perform binary classification. We fine-tune a BERT base model (Devlin et al., 2019) with an additional linear layer on top that maps the hidden state corresponding to the BOS (beginning of a sentence) token to the probability of the positive label. Given the available supervision, we train the architecture to maximize the model likelihood penalized by entropy regularization and weight decay. The final model has an accuracy of 77.1%.[9]

B.2 Question answering

We also evaluate on a task with a more complex sample space: closed-book question answering (QA). Here QA is treated as a sequence-to-sequence problem from question to answer without retrieving nor providing any evidence (Roberts et al., 2020). This, as in FC, emphasises the role of the knowledge acquired in pre-training and encoded in the parameters of the model. For this task, we used the Zero-Shot Relation Extraction (zsRE) dataset by Levy et al. (2017). We prefer zsRE to other popular QA datasets such as SQuAD (Rajpurkar et al., 2016), Natural Questions (Kwiatkowski et al., 2019) or TriviaQA (Joshi et al., 2017) because it is annotated with human-generated question paraphrases that we can use to evaluate our model's robustness to semantically equivalent inputs. zsRE is specifically constructed not to have relation overlaps between training and test (*i.e.* it is zero-shot). We re-split the dataset to have the same distribution in training and test splits—we are not interested in zero-shot specifically, so we avoid the additional complexity it entails. The original zsRE dataset has 147,909 training and 3,724 validation instances respectively. After re-splitting and employing all paraphrases, we have 244,173 training and 27,644 validation instances respectively. For this task, we fine-tune a BART base model (Lewis et al., 2020) with a standard seq2seq objective, *i.e.*, maximizing the model likelihood given the observed output sequence (Sutskever et al., 2011, 2014) and regularized with dropout (Srivastava et al., 2014) and label smoothing (Szegedy et al., 2016). The final model has an accuracy (exact match between model prediction and gold standard) of 22.1%.[10]

B.3 Generating alternative predictions

Generation of alternative predictions is task-dependent as it requires producing a plausible substitute target for a given input—*e.g.*, if we need to edit the knowledge about a head of a state, a plausible substitute label should be a person, not a random (even if well-formed) string. Fact-Checking is straightforward: we simply flip the label, as it is binary classification. For QA, we exploit high-probability outcomes under the model distribution as a proxy to plausible revisions. In particular, we pick all hypotheses enumerated via beam search except the top-1.[11]

[9] This is comparable with what reported by Petroni et al. (2021) for a larger BART model.

[10] This is more than reported by Petroni et al. (2021) on the original split of zsRE. That is because the original split aims at zero-shot evaluation, while we have an overlap of relation types between training and validation sets.

[11] This does not always guarantee that the alternative predictions have the same semantic type as the original one, but it is likely since the model assigns high probability to them.

B.4 Semantically equivalent inputs

We would like the updated model to be consistent for semantically equivalent inputs (see $\mathcal{P}^x$ in Section 2 and 4) as opposed to *just* learning a new specific and isolated datapoint. This consistency is indicative of an effective editing mechanism that taps into the knowledge stored in the model. However, not all datasets come with paraphrases of its inputs (*e.g.*, in our case FEVER does not come with paraphrases and zsRE only has paraphrases for 30% for the dataset). To this end, we generate semantically equivalent inputs using round-trip translation (Sennrich et al., 2016; Wieting and Gimpel, 2018). We employ English-to-German and German-to-English Transformer models from Marian Neural Machine Translation (MarianNMT; Junczys-Dowmunt et al., 2018) provided by Huggingface Transformers (Wolf et al., 2020). We use beam search with beam size 5 to obtain 25 paraphrases. From this set, we exclude any candidate paraphrase $\hat{x}$ of x for which the prediction $\hat{y}$ supported by $f(\hat{x};\theta)$ does not match the prediction y supported by $f(x;\theta)$. This filtering ensures that, according to the current model, all paraphrases have the exact same prediction.

B.5 Architecture details

The original models we want to modify are a BERT base model (Devlin et al., 2019) and a BART base model (Lewis et al., 2020) for fact-checking and question answering respectively. They are both Transformer based models with 12 layers each and hidden size of 768. BERT has 12 heads, where BART has 16. They have 110M and 139M parameters respectively. BERT has a vocabulary size of 30,522 where BART has 50,265.

KNOWLEDGEEDITOR has a small single-layered bidirectional-LSTM with input size 768 and hidden size of 128. The FFNN that condenses the LSTM states follows a [256, tanh, 1024] architecture where the 5 FFNN have all a [1024, tanh, d] architecture where d depends on the weight to modify. In our experiments, we do not use our model to modify biases, layer norms, word and positional embeddings of LMs. Overall, KNOWLEDGEEDITOR has 54M and 67M parameters for BERT and BART respectively.

B.6 Training details

The original models which we want to modify are trained with a batch size of 256 using Adam (Kingma and Ba, 2015) (learning rate of 3e-5) with weight decay (1e-2) and a linear schedule with warm-up (50k total number of updates and 500 warm-up updates). We trained for a maximum of 20 epochs and employ model selection using accuracy on the validation set.[12]

KNOWLEDGEEDITOR models are trained with a batch size of 1024 for FC and 256 for QA using Adam (learning rate of 3e-4 for the parameters and 1e-1 for the Lagrangian multiplier) with weight decay (1e-2) and a linear schedule with a warm-up (200k total number of updates and 1k warm-up updates). We trained for a maximum of 200 epochs and employ model selection using overall accuracy (success rate and retain accuracy) on the validation set (approximated using mini-batches).[13] The margin for the $\mathcal{C}_{KL}$ is annealed between 1e-1 and 1e-3 for the fact-checking model, and between 1e-3 and 1e-5 for the BART question answering model. For the sequence-to-sequence loss, we employ a cross-entropy loss with label smoothing of 0.1.

C Additional Results

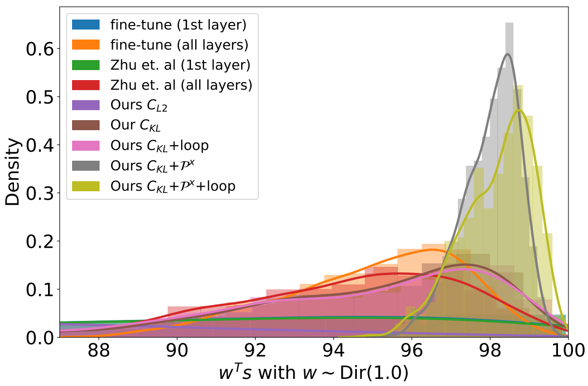
Update Analysis During preliminary experiments, we studied a version of our hyper-network that did not exploit gradient information (see Equation 3). Without gradient information, on FC the models converged $\approx$10 times slower to reach the same accuracy and did not converge for QA (*i.e.*, the model was not able to get $> 75\%$ success rate and $> 50\%$ retain accuracy). That suggest that the gradients are helpful and actually used by our hyper-network but should not used directly, without a modification. To better show this, in Table 2 we report correlations between different update methods and the gradient in terms of cosine similarities between updates. Naturally, fine-tuning and the gradient are highly correlated, but our method (with and without additional paraphrases supervision), poorly correlates with the others. Low cosine similarity can be due to two factors i) the model indeed projects the gradient to a different and more ‘knowledge preserving’ direction, or ii) the parameter space is so large that cosine similarity gets to zero very quickly, not revealing the genuine underlying similarity.

[12]We trained on 4 Nvidia Titian X 12GB which take approximately 10 minutes for FC and 3 hours for QA.

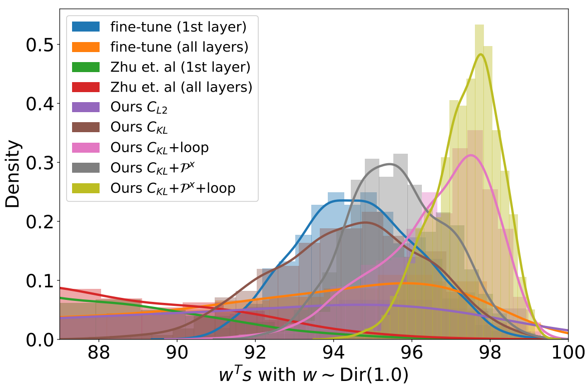
[13]We trained on 4 Nvidia Titian X 12GB which take approximately 1 day for FC and 3 days for QA.

	$\nabla_\theta \mathcal{L}$	Fine-tune	$\mathcal{C}_{KL}$	$\mathcal{C}_{KL} + \mathcal{P}^x$
$\nabla_\theta \mathcal{L}$	1.000	0.451	-0.018	-0.025
Fine-tune	0.451	1.000	-0.010	-0.011
$\mathcal{C}_{KL}$	-0.017	-0.010	1.000	0.183
$\mathcal{C}_{KL} + \mathcal{P}^x$	-0.021	-0.011	0.183	1.000

Table 2: Average cosine similarities between different update methods and the gradient for the update as well. Fine-tuning is applied to all layers.

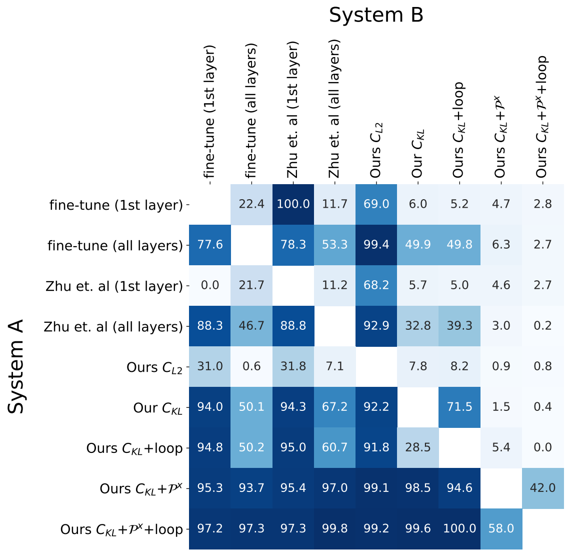


(a) Fact-checking.

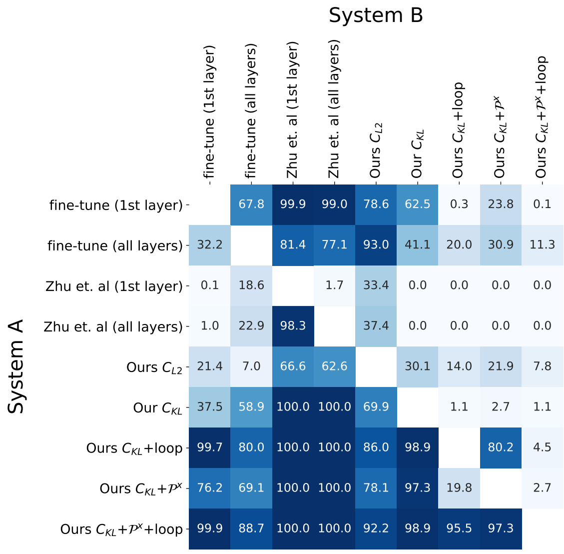


(b) Question answering.

Figure 5: Probability distributions of weighted sum of metrics according to 1k random assignments sampled from a Dirichlet distribution (with $\alpha = 1$—see all values in Table 1). Sampling weights allows to interpret the score in a probabilistic way. KNOWLEDGEEDITOR (with different variants) presents distributions that are more skewed towards a high score (100) indicating that it is highly likely that when assigning some weights to the metrics, the weighted sum will be in favour of our method. *Better view with colors.*



(a) Fact-checking.



(b) Question answering.

Figure 6: Probability that *system A* is better than *system B* according to a weighted sum of metrics (see individual values in Table 1) sampling mixing coefficients 1,000 times from a Dirichlet distribution (with $\alpha = 1$ to cover a diverse spectrum of metric combinations). The probability that KNOWLEDGEEDITOR (with $\mathcal{C}_{KL}$ + $\mathcal{P}^x$ + loop) is better than competing systems is high ($> 97\%$ for FC and $> 88\%$ for QA) indicating that it is highly likely that when assigning some weights to the metrics, the weighted sum will be in favour of our method. *Better view with colors.*