E.2 [GPT-2 XL only] Knowledge Neurons (KN)

The method by Dai et al. (2022) first selects neurons that are associated with knowledge expression via gradient-based attributions, and then modifies $\mathrm{mlp}_{proj}^{(l)}$ at the rows corresponding to those neurons by adding scaled embedding vectors. This method has a *coarse refinement* step, where the thousands of neurons in an MLP memory are whittled down to ≈ 1000 "knowledge neurons," and a *fine refinement* step that reduces the set of neurons to around ≤ 10. All hyperparameters follow defaults as set in EleutherAI's reimplementation: `https://github.com/EleutherAI/knowledge-neurons`.

E.3 [GPT-2 XL only] Knowledge Editor (KE)

De Cao et al. (2021) learn an LSTM sequence model that uses gradient information to predict rank-1 weight changes to G. Because the official code does not edit GPT-2, we use Mitchell et al. (2021)'s re-implementation in their study. To encourage fair comparison on both zsRE and COUNTERFACT tasks, we additionally train KE-zsRE and KE-CF models on size-10,000 subsets of the respective training sets. Hyperparameters for training are adopted from the given default configuration. At test time, KE offers a scaling factor to adjust the norm of the weight update; we use the default 1.0.

E.4 [GPT-2 XL, GPT-J] Model Editor Networks with Gradient Decomposition (MEND)

Mitchell et al. (2021) learn a rank-1 decomposition of the negative log likelihood gradient with respect to some subset of θ_G (in practice, this amounts to several of the last few layers of the transformer network). Again, for fair comparison, we train new versions of MEND (MEND-zsRE, MEND-CF) on the same sets that KE-zsRE and KE-CF were trained on. Similar to KE, hyperparameters for training and test-time inference are adopted from default configurations.

E.5 [GPT-2 XL, GPT-J] Rank-One Model Editing (ROME)

ROME's update (Section 3.1) consists of key selection (Eqn. 3), value optimization (Eqn. 4), and v insertion (Appendix A). We perform the intervention at layer 18. As Figure 1k shows, this is the center of causal effect in MLP layers, and as Figure 3 shows, layer 18 is approximately when MLP outputs begin to switch from acting as keys to values.

Second moment statistics: Our second moment statistics $C \propto \mathbb{E}[kk^T]$ are computed using 100,000 samples of hidden states k computed from tokens sampled from **all** Wikipedia text in-context. Notice that sampling is not restricted to only special subject words; every token in the text is included in the statistic. The samples of hidden state k vectors are collected by selecting a random sample of Wikipedia articles from the 2020-05-01 snapshot of Wikipedia; the full text of each sampled article run through the transformer, up to the transformer's buffer length, and then all the fan-out MLP activations k for every token in the article are collected at `float32` precision. The process is repeated (sampling from further Wikipedia articles without replacement) until 100,000 k vectors have been sampled. This sample of vectors is used to compute second moment statistics.

Key Selection: We sample 20 texts to compute the prefix (x_j in Eqn. 3): ten of length 5 and ten of length 10. The intention is to pick a k_* that accounts for the different contexts in which s could appear. Note that we also experimented with other x_j sampling methods:

- **No prefix**. This baseline option performed worse ($\mathrm{S}' = 86.1$ compared to $\mathrm{S} = 89.2$).
- **Longer prefixes**. Using $\{$ ten of length 5, ten of length 10, and ten of length 50 $\}$ did not help performance much ($\mathrm{S}' = 89.3$).
- **More same-length prefixes**. Using $\{$ thirty of length 5 and thirty of length 10 $\}$ did not help performance much ($\mathrm{S}' = 89.2$).

Value Optimization: v_* is solved for using Adam with a learning rate of 0.5 and 1.5×10^{-3} weight decay. The KL divergence scaling factor, denoted λ in Eqn. 4, is set to 1×10^2. The minimization loop is run for a maximum of 20 steps, with early stopping when $\mathcal{L}(z)$ reaches 5×10^{-2}.

The entire ROME edit takes approximately $2\mathrm{s}$ on an NVIDIA A6000 GPU for GPT-2 XL. Hypernetworks such as KE and MEND are much faster during inference (on the order of $100\mathrm{ms}$), but they require hours-to-days of additional training overhead.