We thank the reviewers for their time and comments. Due to space limitations we could only address major points, but we'll try to reflect all advice in future revisions.

Response to Reviewer #1

Thank you for the advice and comments! In the next revisions we are planning to add other more diverse datasets to present stronger experimental evidence.

Response to Reviewer #2

Q1. Pre-clustering baseline. We have run a pre-clustering baseline on MSCOCO to respond to your concern. Specifically, we run k-means to obtain 778 clusters (corresponding to our setting $\alpha = 0.1$) of training sentences embedded by Sentence-BERT [1], then for each cluster identify one prototype whose embedding is the closest to the cluster centroid. We use these 778 prototypes and a uniform prior over them, and train with the same objective as Guu et al. (note that prototypes are retrieved based on Sentence-BERT embeddings and the edit representation is the same as ours). The importance-weighting approximated PPL is **19.5 against 18.6** from our model, which shows that the sparse prototypes selected by pre-clustering is sub-optimal compared to our method. We will add these details to the paper.

[1] Nils Reimers et al. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. EMNLP 2019

Q2. Setting prior over t to be truncated mean of $q_\lambda(\theta)$ (on training data) at test time? Yes, it is correct.

Q3. Explanation about PPL improvement over NLM. We think the PPL improvement reflects that the prototype-style model provides an easier way to generate certain examples than generating from scratch. Interpolation with NLM also leads to large improvement as observed by Guu et al. as well due to a ensembling effect of combining the two models. We will try to add quantitative detailed analysis as suggested by the reviewer to understand the behaviour.

Response to Reviewer #3

Q1. Sparsity may be leading to improvements even where no subsampling is needed? When we run a dense baseline on MSCOCO with $\alpha = 10$ and 35K (90%) active prototypes, PPL is 19.9 vs. 18.6 with 778 prototypes, which implies that sparsity may lead to improvement. We'll include results from more α settings in the next revision.

Q2. Inaccurate statement about posterior collapse. This is our oversight and thank you for pointing out! We meant that non-zero KL often practically leads to non-collapsed posterior as shown in VAE text modeling literature (Xu & Durrett, 2018; Li et al., 2019). We'll make sure to correct the claims properly in next revision.

Q3. Changing the support at test time? This is a great point. It is possible to retrieve from a new prototype set at test time using the trained sentence encoder as in Equation (9). The prior over these new prototypes can be set by either using mean of posterior $q_\lambda(\theta)$ by running inference once on the training data, or just using a uniform prior. However, there is no sparsity guarantee in this case and we may lose the efficiency advantages. Also, your understanding is correct that the "speedup" at test time comes from concentrating on a small subset of the potentially sub-sampled training set.

Q4. Interpolation with another NLM. We use linear interpolation: $p(x) = \lambda p_{\text{proto}}(x) + (1 - \lambda) p_{\text{NLM}}(x)$. The coefficient λ is 0.1 for prototype-based model, the same value as used in Guu et al. for a fair comparison.

Response to Reviewer #4

Q1. Other time-efficient retrieval options. We think other methods that improve retrieval time efficiency without changing the prototype support are complementary to our approach; we aim to improve *both* memory and time efficiency by reducing prototype size. This is particularly important in the current trend towards huge training data and models.

Q2. Pre-clustering examples? Please see Q1 of Reviewer $\#2$ for comparison details: sparse prototypes identified by heuristic pre-clustering are not as effective as those learned by our approach.

Q3. Sparsity may not be needed for good performance? We agree, but our focus is mainly on improving memory efficiency of non-parametric models at test time without losing performance. Also, the statement that "at inference time we only retrieve one for generation" is not correct – theoretically we marginalize over all latent template variables, and practically we use multiple template samples (line 219) per test example to approximate the marginal likelihood.

Q4. Restricting prototype size goes against the spirit of non-parametric modeling? We don't agree on this. Strictly speaking, we just encourage sparsity for the non-parametric model instead of restricting prototype size. Much information in huge training datasets is redundant, and even for traditional non-parametric kNN models that were well-studied, data reduction is one of the most important problems for work with huge data sets (kNN Wikipedia page).

Q5. Why does our method have better PPL than neural editor? We think the PPL improvement over the neural editor is mainly from *learnable* prototypes vs. the fixed prototypes used by Guu et al. This is further supported by our pre-clustering baseline results in Q1 of Reviewer $\#2$.

Q6. Why does the PPL go down with sparser prototypes? The PPL improvement may be related to inductive bias of our model which is encouraged to focus on a sparse subset of prototypes. This might cause the decoder to find more useful ways to edit. Yet it is not always the case, for example, PPL on Yelp Medium with 77 prototypes goes up to 63.6.