A Fresh Take on Stale Embeddings: Improving Dense Retriever Training with Corrector Networks

Nicholas Monath [*1] **Will Grathwohl** [*1] **Michael Boratko** [1] **Rob Fergus** [1] **Andrew McCallum** [1] **Manzil Zaheer** [1]

Abstract

In dense retrieval, deep encoders provide embeddings for both inputs and targets, and the softmax function is used to parameterize a distribution over a large number of candidate targets (e.g., textual passages for information retrieval). Significant challenges arise in training such encoders in the increasingly prevalent scenario of (1) a large number of targets, (2) a computationally expensive target encoder model, (3) cached target embeddings that are out-of-date due to ongoing training of target encoder parameters. This paper presents a simple and highly scalable response to these challenges by training a small parametric *corrector network* that adjusts stale cached target embeddings, enabling an accurate softmax approximation and thereby sampling of up-to-date high scoring "hard negatives." We theoretically investigate the generalization properties of our proposed target corrector, relating the complexity of the network, staleness of cached representations, and the amount of training data. We present experimental results on large benchmark dense retrieval datasets as well as on QA with retrieval augmented language models. Our approach matches state-of-the-art results even when no target embedding updates are made during training beyond an initial cache from the unsupervised pre-trained model, providing a 4-80x reduction in re-embedding computational cost.

1. Introduction

The softmax function, paired with deep neural encoder models, is often the parameterization of choice for discrete distributions over many targets such as in classification (Logeswaran et al., 2019; Yu et al., 2022), retrieval (Reddi et al., 2019; Xiong et al., 2020), or reinforcement learning (Dulac-Arnold et al., 2015; Gottipati et al., 2020). This approach, often called a "dual encoder," employs two separate deep networks, one to map an input to a fixed dimensional vector, another to map targets to the same vector space. We then compute softmax logits as the inner product of an input vector to each target vector (Gillick et al., 2019; Karpukhin et al., 2020; Xiong et al., 2020).

With the typical softmax cross-entropy loss, exact training of the parameters of these two encoder networks would involve using the current parameters to compute the logits for all targets, requiring running the target encoder on all targets at every step of training. Of course, this far-too-burdensome approach is not used in practice. Instead, various approximations have been developed (Reddi et al., 2019; Rawat et al., 2020; Lindgren et al., 2021; Xiong et al., 2020; Monath et al., 2023). The typical approximation computes a truncated softmax on a sampled subset of targets. These approaches store a *cache* of "stale" encoded representations of targets and uses the stale, cached representations to draw samples from the softmax-parameterized distribution during training (Lindgren et al., 2021; Izacard et al., 2022). Previous work has used these stale representations amidst other approximations such as index structures (Xiong et al., 2020; Monath et al., 2023), kernel-methods (Rawat et al., 2019), and focusing training on subsets of targets (Reddi et al., 2019). However, inevitably, the staleness of the target embeddings causes training regret.

In this work, we present a simple, general purpose method for addressing staleness in softmax-parameterized categorical distributions that is scalable enough to be updated at every step of training. Our approach improves upon an existing stale approximation using a learned *target corrector network*. The target corrector network, inspired by recent work on training continuous energy-based models (Han et al., 2020; Grathwohl et al., 2020; 2021), is a small parametric model that accounts for the discrepancy between the stale approximation and unnormalized logits from the true distribution. By learning to improve upon the stale approximation, the target corrector network can be used to produce a more accurate approximation to the target distribution. We further extend beyond training large output space classifiers to latent variable retrieval augmented language models.

[*]Equal Contribution. Order determined by coin flip. [1]Google DeepMind. Correspondence to: Will Grathwohl <wgrathwohl@google.com>, Nicholas Monath <nmonath@google.com>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

In summary, the contributions of this paper are:

Methodological (§3) - We describe a novel training procedure for large output space models. It is based on approximating softmax-parameterized categorical distributions by using a parametric target corrector network that learns to improve stale approximations of logits.

Theoretical (§4) - We analyze the generalization properties of the corrector networks in terms of the discrepancy between the stale approximation and the true distribution, the complexity of the network, and the amount of training data.

Empirical (§5) - We evaluate our approach in training both dense retrieval models and latent variable retrieval augmented language models. Our approach matches the performance of much more computationally intensive approaches at a fraction of the computational expense.

2. Background

Softmax Given an input point x, a distribution over a set of N *targets*, $\mathcal{Y}$, parameterized by the softmax function is:

$$P(y|x) = \frac{\exp(\beta s_{x,y})}{Z_x \triangleq \sum_{y' \in \mathcal{Y}} \exp(\beta s_{x,y'})}, \tag{1}$$

where β is the temperature. In this paper, we focus on applications in retrieval and latent variable models. For example, in Natural Questions (Kwiatkowski et al., 2019), x refers to a question and targets, y, correspond to Wikipedia passages.

Dual-Encoders We compute the unnormalized logits, $s_{x,y}$, using a factorized representation. Deep parametric models, *dual-encoders*, map the input, x, and target, y, to D-dimensional vectors, denoted $f(x;\Theta)$, and $g(y;\Theta)$:

$$s_{x,y} = \langle f(x;\Theta), g(y;\Theta) \rangle. \tag{2}$$

Training For a task-specific loss, $\mathcal{L}$, such as cross-entropy, dual-encoder parameters are optimized by gradient descent (Rawat et al., 2019). However, exact computation of the normalizing constant, Z_x, is typically intractable during training, since it would require computing $g(y)$ for millions or billions of targets. Instead of $P(y|x)$ in $\mathcal{L}$, a tractable (yet biased) approximation is to optimize the *truncated* softmax, $\tilde{P}(y|x)$, including only a subset of targets $S(\mathcal{Y}) \subset \mathcal{Y}$:

$$\tilde{P}(y|x) = \frac{\exp(\beta s_{x,y})}{\sum_{y' \in S(\mathcal{Y})} \exp(\beta s_{x,y'})}, \tag{3}$$

Uniform Sampling Approximation A simple approach is to define $S(\mathcal{Y})$ to be a uniformly sampled subset of $\mathcal{Y}$ (Karpukhin et al., 2020). The method's bias decreases with more samples. However, since the samples are uniform, a large number of samples may be required.

Top-K / Similarity-based Sampling Approximations We can instead use an informed strategy using $g(y)$ that would select higher probability targets by sampling using similarity scores via Gumbel-Max (Lindgren et al., 2021), or using the top-k targets in terms of inner product (Xiong et al., 2020). Work has considered efficient approximations to find these top k targets without having to compute $g(y)$ for all $y \in \mathcal{Y}$ (Xiong et al., 2020; Monath et al., 2023).

Initialization We initialize the parameters of the dual encoders, Θ, using pre-trained models, such as pre-trained language models, T5 and GTR (Devlin et al., 2019; Raffel et al., 2020; Ni et al., 2022).

Stale Cached Representations When we are training the parameters, Θ, each target's vector according to $g(y;\Theta)$ changes at each step of training. Therefore, a commonly used approach is to define an approximation, $g'(y)$, that is a lookup for a "stale" cached embedding for the given target. The stale embedding comes from running the target encoder at a particular time step t, of training, and caching the result, i.e., $g(y, \Theta_t)$ in a buffer, $B \in \mathcal{R}^{|\mathcal{Y}| \times D}$, i.e. $g'(y_i) \triangleq B_{y_i}$. To find the top-$k$ targets for input x we compute approximate logits $B^T f(x) \in \mathcal{R}^{|\mathcal{Y}|}$ and select the top-k targets to define $S(\mathcal{Y})$. Even before training, we can use the pre-trained model to produce embeddings for all targets $B_{y_i} = g(y_i; \Theta_0)$. While B may seem large, this is considerably more efficient than exact computation and is possible, on accelerators, for $|\mathcal{Y}|$ in the tens of millions.

The bias of this approach (and subsequent degradation in performance) depends on the *staleness* or drift of the embeddings, i.e., $||B_{y_i} - g(y_i)||$ which will increase as we update the parameters of $g(y)$. This can be mitigated by recomputing B periodically throughout training (at notable cost). This approach of periodically recomputing has been used (Guu et al., 2020; Izacard et al., 2022; Monath et al., 2023), but there is still much room for improvement.

3. Improving Training with Target Correctors

Our proposed approach builds upon these stale buffer approximations by using an additional parametric model. The additional model aims to improve upon the stale $g'(y)$ to yield a better approximation of $g(y)$.

We refer to this additional parametric model as a *target corrector network*, $h(\cdot;\Psi)$ or simply $h(\cdot)$ when the parameters Ψ are not pertinent. This target corrector network takes as input the existing stale vector embedding, $g'(y)$, and yields the following approximation of the softmax function:

$$P_h(y|x) \propto \exp(\beta \langle f(x), h \circ g'(y) \rangle). \tag{4}$$

With significantly fewer parameters than a typical dual-encoder, i.e., $|\Psi| \ll |\Theta|$, this small parameteric model is efficient enough to provide approximately fresh representations of every target at every training step. The target corrector network presents interesting research questions regarding whether the network can obviate the need for re-embedding, what kinds of staleness or drift can be effectively modeled, and how much training data is required.

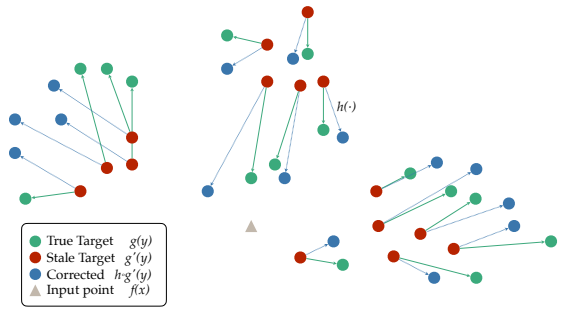

Figure 1. **Target Corrector Networks**. The corrector network, $h(\cdot)$, moves the approximate representations of targets, $g'(\cdot)$ to be closer to their true positions, $g(\cdot)$. The corrector network is trained to approximate how the targets are transformed from $g'(\cdot)$ to $g(\cdot)$.

Warmup: Training the corrector network in isolation We begin by considering how we would train only the parameters of the target corrector network, independently of the dual-encoders $f(x)$ and $g(y)$. Afterwards, we present an algorithm for jointly training the target corrector network and the dual-encoders. To train the parameters Ψ of the corrector network, $h(\cdot;\Psi)$, we collect training examples of input data points x_i, the exact target embeddings $g(y)$, and stale embeddings $g'(y)$ for a subset of targets $S(\mathcal{Y})_i \subset \mathcal{Y}$, i.e., $\{(f(x_i), g(y_b), g'(y_b)) \mid y_b \in S(\mathcal{Y})_i\}$.

We consider two loss functions for training h: the mean-squared error between representations given by $g(y)$ and the corrected representations $h \circ g'(y)$ (Eq. 5) and the cross entropy loss between the truncated softmax using $g(y)$ and truncated softmax using $h \circ g'(y)$ (Eq. 6):

$$\ell_{MSE}(y_i) = ||g(y_i) - h \circ g'(y_i;\Psi)||_2 \tag{5}$$

$$\ell_{\mathrm{d}}(y_i) = \log \tilde{P}(y|x) - \log \tilde{P}_h(y|x) \tag{6}$$

$$\tilde{P}_h(y|x) = \frac{\exp(\beta\langle f(x), h \circ g'(y;\Psi)\rangle)}{\sum_{y' \in S(\mathcal{Y})_i} \exp(\beta\langle f(x), h \circ g'(y';\Psi)\rangle)}.$$

where $\tilde{P}(y|x)$ is the truncated softmax $g(y)$ (Eq. 3). The mean-squared error loss directly tries to match the target encoder model's embeddings. The cross-entropy loss down-weights the importance of targets y which do not contribute substantial probability to $P(y|x)$ and allows for greater use of model capacity. The parameters of the target corrector networks are optimized using gradient descent. Empirically, we find the cross-entropy objective to perform slightly better (Table 1) and focus the presentation on cross-entropy.

Jointly Training Corrector Networks & Dual-Encoders We present a method (Algorithm 1) for simultaneously training dual-encoders for a given task (e.g., retrieval or equivalently large output-space classification) and the target corrector network. The training algorithm will optimize both the parameters of the target corrector network and additionally use the corrector network to approximate the softmax. Each step consists of: (1) using the corrector network to provide an approximately updated representation of every target, (2) picking a subset of targets for the truncated softmax using the output of the corrector network, (3) computing a task loss for the dual-encoder models and loss for the corrector networks, (4) updating, according to their respective losses, the parameters for both the dual-encoders and the corrector networks using gradient descent.

In more detail, we are given task training data, $X = \{(x_1, y_1), \ldots, (x_m, y_m)\}$. We are given a task loss function $\mathcal{L}$ and a corrector network loss ℓ. The dual-encoder models are $f(x), g(y)$ and their initial parameters are Θ_0. Prior to the first training step, we instantiate a buffer of the targets' representations, $B_y = g'(y) = g(y;\Theta_0)$. We will avert the need for the expensive updating of the buffer by re-embedding targets with the target encoder. In each step, we sample a training point and label pair x_i, y_i from X. We apply the target corrector network to all of the stale representations in the buffer to obtain $h \circ g'(y)\ \forall y \in \mathcal{Y}$. This computation does not require running a dual-encoder; we use the cached buffer representation of each target as input to the corrector network. The corrector network is typically a two-layer MLP and hence efficient enough to be used in this way. With these representations from $h(\cdot)$, we sample (or select exact top-k) targets according to $P_h(y|x)$ (Eq. 4) to form a subset of targets $S_{x_i}(\mathcal{Y})$ for the truncated softmax.

Given this subset, we compute the task and correction losses and update their respective model parameters. First, we compute the task loss, which is cross-entropy. The task loss will only be used to update the parameters of the dual-encoders, Θ, not the parameters of the target corrector network. We compute the truncated softmax $\tilde{P}(y|x) \propto \exp(\beta\langle f(x), g(y)\rangle)$ (Equation 3). We define a one-hot $P^\star$ according to the training data label y_i. We compute the task specific loss $\mathcal{L}$ as a function of $\tilde{P}$ and $P^\star$, and update the dual encoder parameters via gradient descent $\Theta \leftarrow \Theta - \eta\nabla_\Theta\mathcal{L}$.

Next, we will use the same sample of targets $S_{x_i}(\mathcal{Y})$ to compute the target corrector network's loss and parameter update. Importantly, this will only update the parameters of the target corrector network, Ψ, not the parameters of the dual-encoders. Here we describe the use of the cross-entropy loss. However, an analogous update procedure could be used for other loss functions. We compute the truncated softmax according to the target corrector network's output: $\tilde{P}_h(y|x) \propto \exp(\beta\langle f(x), h \circ g'(y)\rangle)$. We then compute the target corrector network loss, ℓ, cross-entropy, which tries to align two truncated distributions $\tilde{P}_h$ and $\tilde{P}$. The target corrector network's parameters are updated by gradient descent $\Psi \leftarrow \Psi - \eta\nabla_\Psi\ell$.

Training the target corrector network, which has only a small number of parameters, is much less computationally inten-

sive to train than the dual-encoder model. Furthermore, we are given "for free" the representations $g(y)$ since they are used to compute $\tilde{P}$ for the task loss. These representations can then easily be re-used for training the corrector.

The training procedure is summarized in Algorithm 1. At prediction time, the corrector network is not used, instead the trained dual-encoder $g(y, \Theta)$ is used.

Algorithm 1 Training with target corrector networks

Data: Training data X, Targets $\mathcal{Y}$, Input encoder $f(\cdot)$, Target encoder $g(\cdot)$, Approximate target encoder $g'(\cdot)$ (buffer B), target corrector network $h(\cdot)$, temperature β, task loss $\mathcal{L}$, target corrector network loss ℓ, learning rate η, number of truncated samples k

while Training **do**

  Sample training data $(x_i, y_i) \sim X$

  Compute $h \circ g'(y)$ for all $y \in \mathcal{Y}$ using the buffer B

  Set $S_{x_i}(\mathcal{Y})$ using $\exp(\beta f(x_i)^T h \circ g'(y))$ via top-k

  Include supervised label $S_{x_i}(\mathcal{Y}) \leftarrow S_{x_i}(\mathcal{Y}) \cup \{y_i\}$

  Define $\tilde{P}(y|x_i) = \frac{\exp(\beta f(x_i)^T g(y))}{\sum_{y' \in S_{x_i}(\mathcal{Y})} \exp(\beta f(x_i)^T g(y'))}$

  Define $\tilde{P}_h(y|x_i) = \frac{\exp(\beta f(x_i)^T h \circ g'(y))}{\sum_{y' \in S_{x_i}(\mathcal{Y})} \exp(\beta f(x_i)^T h \circ g'(y'))}$

  Define $P^\star$ to be a one-hot vector for y_i.

  Compute task loss $\mathcal{L}$ using $\tilde{P}$ and $P^\star$

  Compute correction loss ℓ using $\tilde{P}$ and $\tilde{P}_h$

  Update dual-encoder parameters $\Theta \leftarrow \Theta - \eta \nabla_\Theta \mathcal{L}$

  Update corrector network parameters $\Psi \leftarrow \Psi - \eta \nabla_\Psi \ell$

end

3.1. Latent Variables in Retrieval Augmented Models

Retrieval augmented language models (RLMs) typically consist of two major architectural components, a *retriever* model (e.g., a dual-encoder) and a generative language model or *reader* model (Guu et al., 2020; Izacard & Grave, 2020; Izacard et al., 2022). The input to a retrieval augmented language model is a natural language text sequence, x. This input text will be encoded using a dual-encoder retrieval model, $f(x)$. Retrieval will be performed over a corpus of targets, $\mathcal{Y}$, returning k targets relevant to x, denoted $S_x(\mathcal{Y})$. The reader model takes as input the retrieved targets, $S_x(\mathcal{Y})$, and the text x, and generates text.

Concretely, in our experiments, the input text x is a question. The retrieval corpus contains targets y corresponding to passages in Wikipedia. The reader model takes as input the question and retrieved passages and generates a short answer to the question. We present the remainder of the section with this question-answering task in mind.

RLMs can be formalized as latent variable models. The softmax function is used to parameterize the distribution over a discrete latent variable, which corresponds to the retrieved targets. We use a to refer to the generated sequence of text, i.e., the generated answer:

$$P(a|x) = \sum_{y \in S_x(\mathcal{Y})} P(a|y, x) P(y|x). \tag{7}$$

$P(a|y, x)$ is an autoregressive language model. $P(y|x)$ is computed by the softmax with logits from Equation 2 using the encoder models $f(x)$ and $g(y)$.

When training RLMs, we receive supervision in the form of question, answer pairs, e.g., $x_i, a_i \sim X$. We do not receive supervision on which targets $S_x(\mathcal{Y})$ should be retrieved. We will learn the parameters of both the reader model and retriever model using these supervised question/answer pairs.

To train the reader and retriever model, we use perplexity distillation (Izacard et al., 2022) for retriever loss and negative log-likelihood for the reader loss. Perplexity distillation is computed as the cross-entropy between two truncated distributions, one being the retriever's $\tilde{P}(y|x)$ (Equation 3) and the other using the reader model to provide a soft-relevance label for each target in $S_x(\mathcal{Y})$:

$$P_a(y|x) = \frac{P(a|y, x)}{\sum_{y' \in S_x(\mathcal{Y})} P(a|y', x)}. \tag{8}$$

In words, $P_a(y|x)$ normalizes the likelihood scores of the reader model generating the correct answer text when conditioned on the given retrieved target y. The reader's loss function, negative-log likelihood is simply computed using the supervised answer text. The two losses are averaged and parameters optimized with gradient descent.

To facilitate efficient training, we use our proposed target corrector network to select the subset of retrieved targets $S_x(\mathcal{Y})$ used at training time. This is done in the same way as in Algorithm 1, i.e., we pick a subset of k targets $S_x(\mathcal{Y})$ for x according to $\exp(\beta f(x)^T h \circ g'(y))$ via top-k or Gumbel-Max sampling. We can make simple modifications to Algorithm 1, which are presented in Algorithm 2 to train the RLM. We compute two task-specific losses (perplexity distillation, negative log-likelihood) and optimize both the reader and retriever parameters. We use cross-entropy to train the corrector, which is again only used at training time. At prediction time, the trained retriever model is used.

4. Analysis

We will explore the generalization of the proposed target corrector network in terms of unseen targets for a particular input data point, and will show the relationship between generalization error, the complexity of the target corrector network h, and the discrepancy of the stale representations, g', and true representations g. All proofs are in Appendix A.

Let $\ell : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ is a loss function for the target corrector network (Eq. 5 & 6). For any point x, consider the distribution given by the softmax using stale approximation g':

Algorithm 2 Training RLMs with corrector models

Data: Training data X, Targets $\mathcal{Y}$, Retriever and Reader Parameters Θ, Correction Model parameters Ψ, Input encoder $f(\cdot)$, Target encoder $g(\cdot)$, Approximate target encoder $g'(\cdot)$ (buffer B), corrector model $h(\cdot)$, temperature β, retriever loss $\mathcal{L}$, reader loss $\mathcal{L}'$, corrector model loss ℓ, learning rate η, number of truncated samples k

while Training **do**

  Sample training data $(x_i, a) \sim X$

  Compute $h \circ g'(y)$ for all $y \in \mathcal{Y}$ using the buffer B

  Set $S_{x_i}(\mathcal{Y})$ using $\exp(\beta f(x_i)^T h \circ g'(y))$ via top-k

  Define $\tilde{P}(y|x_i) = \frac{\exp(\beta f(x_i)^T g(y))}{\sum_{y' \in S_{x_i}(\mathcal{Y})} \exp(\beta f(x_i)^T g(y'))}$

  Define $\tilde{P}_h(y|x_i) = \frac{\exp(\beta f(x_i)^T h \circ g'(y))}{\sum_{y' \in S_{x_i}(\mathcal{Y})} \exp(\beta f(x_i)^T h \circ g'(y'))}$

  Define $P_a(y|x) = \frac{P(a|y,x)}{\sum_{y' \in S_x(\mathcal{Y})} P(a|y',x)}$.

  Define $P_{\text{LM}}(a|x) = \sum_{y \in S_x(\mathcal{Y})} P(a|y,x)P(y|x)$.

  Compute reader loss $\mathcal{L}'$ using $P_{\text{LM}}(a|x)$

  Compute retriever loss $\mathcal{L}$ using $\tilde{P}(y|x_i)$ and $P_a(y|x)$

  Compute correction loss ℓ using $\tilde{P}$ and $\tilde{P}_h$

  Update retriever & reader params $\Theta \leftarrow \Theta - \eta \nabla_\Theta \frac{\mathcal{L}+\mathcal{L}'}{2}$

  Update corrector network params $\Psi \leftarrow \Psi - \eta \nabla_\Psi \ell$

end

$$\tilde{\mathcal{D}}_Y \triangleq P_{g'}(y|x) = \frac{\exp(\beta \langle f(x), g'(y) \rangle)}{\sum_{y' \in \mathcal{Y}} \exp(\beta \langle f(x), g'(y') \rangle)}, \quad (9)$$

and similarly define $\mathcal{D}_Y \triangleq P_g(y|x)$ as the true distribution, using g (Eq. 1).

We begin by defining three kinds of risk.

Empirical Risk On a set of n-targets $\tilde{\mathcal{S}}_n = \{y_1, ..., y_n\}$ sampled from $\tilde{\mathcal{D}}_Y$, we minimize the empirical risk:

$$R_{\ell,\phi}(\tilde{\mathcal{S}}_n) = \frac{1}{n} \sum_{i=1}^{n} \ell(\phi(y_i), g(y_i)), \quad (10)$$

over a function class $\phi \in \mathcal{F}$.

True Population Risk For generalization error, we are interested in how large the true population risk can become over a function class $\phi \in \mathcal{F}$.

$$R_{\ell,\phi}(\mathcal{D}_Y) = \mathbb{E}_{Y \sim \mathcal{D}_Y}[\ell(\phi(Y), g(Y))], \quad (11)$$

We consider the above quantity because we want to ensure good alignment between $g(y)$ and $\phi(y)$ where there is non-trivial probability mass under the true distribution.

Stale Population Risk The stale population risk is defined analogously to true population risk with $\tilde{\mathcal{D}}_Y$ as the distribution, over a function class $\phi \in \mathcal{F}$:

$$R_{\ell,\phi}(\tilde{\mathcal{D}}_Y) = \mathbb{E}_{Y \sim \tilde{\mathcal{D}}_Y}[\ell(\phi(Y), g(Y))]. \quad (12)$$

Function Classes The function class $\phi \in \mathcal{F}$ is large. We will relate this large function class to a restricted class of functions of the form $h \circ g'$ by leveraging the approximate stale representations, g'. In other words, we restrict $\mathcal{F}$ to $\mathcal{F}^{g'} = \{h \circ g' : h \in \mathcal{H}\}$ where $\mathcal{H}$ represents the simpler function class mapping $\mathbb{R}^d \to \mathbb{R}^d$ which can express the discrepancy between the stale g' and current g.

First, we provide a bound on the gap between the population risk and stale population risk. We formalize this in the following lemma. For ease of notation in this exposition, we define $\mathcal{G}_{\ell,\mathcal{F}}$ as the induced function class: $\mathcal{G}_{\ell,\mathcal{F}} = \{y \mapsto \ell(\phi(y), g(y)) : \phi \in \mathcal{F}\}$.

Lemma 4.1. *Given a target encoder g and its stale approximation g', the gap between the true population risk and stale population risk is bounded in the following way:*

$$R_{\ell,\phi}(\mathcal{D}_Y) - R_{\ell,\phi}(\tilde{\mathcal{D}}_Y) \leq \mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y) \leq \|g - g'\|_1 \quad (13)$$

where $\mathcal{W}$ is the Wasserstein distance. Furthermore, if the approximation g' comes from the same neural model as g with parameters perturbed by u as in aforementioned stale approximation, we have: $\|g - g'\|_1 \leq L\|u\|$ with L as the Lipschitz constant.

Next, we connect stale population risk to the empirical risk.

Lemma 4.2. *Given a target encoder g, its stale approximation g', and a set of n-targets $\tilde{\mathcal{S}}_n = \{y_1, ..., y_n\}$ sampled from $\tilde{\mathcal{D}}_Y$,*

$$R_{\ell,\tilde{\phi}_n}(\tilde{\mathcal{D}}_Y) \leq R_{\ell,\tilde{\phi}_n}(\tilde{\mathcal{S}}_n) + \mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}}), \quad (14)$$

where $\mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}})$ is the Rademacher complexity of $\mathcal{G}_{\ell,\mathcal{F}}$.

Now, we can relate the complexity of function class $\mathcal{F}^{g'}$, number of samples n, and the discrepancy of the true g and stale approximate encoders g':

Theorem 4.3. *For a target encoder, g, its stale approximation, g', and the Rademacher complexity $\tilde{\mathfrak{R}}_n(\mathcal{G}_{\ell,\mathcal{F}^{g'}})$, the true population risk $R_{\ell,\phi}(\mathcal{D}_Y)$ is bounded by the following with probability at least $1 - \delta$:*

$$\begin{aligned} R_{\ell,\phi}(\mathcal{D}_Y) &\leq T_1 + T_2 + T_3 \quad (15) \\ T_1 &= R_{\ell,\tilde{\phi}_n}(\tilde{\mathcal{S}}_n) \\ T_2 &= \mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y) \leq L\|u\| \\ T_3 &= \tilde{\mathfrak{R}}_n(\mathcal{G}_{\ell,\mathcal{F}^{g'}}) + \mathcal{O}\Big(\sqrt{\frac{\log(1/\delta)}{n}}\Big) \end{aligned}$$

Note the following implications of these theoretical results:

1. If the corrector network h is too complicated or there are not enough samples n, then h overfits and T_3, will dominate.

2. If g and g' are very different, then term T_2 will dominate.

3. If $h(\cdot)$ is too simple and we cannot fit the sampled data well, then T_1 will dominate.

We empirically explore some of these trade-offs in §5.3

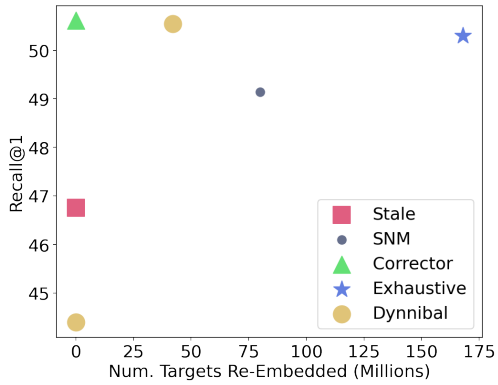

Figure 2. **NQ Test Recall@1**. We show the computational trade-offs between the amount of re-embedding during training and the task performance (GTR initialization). Our proposed target corrector approach achieves matching task performance at a fraction of the computational expense.

5. Experiments

We evaluate training using target corrector networks in two settings: supervised dense retrieval and retrieval augmented language models. We further investigate the properties of the target corrector networks in synthetic experiments. In summary, the experiments investigate whether training strategies can effectively obviate the need to keep cached buffers of targets up-to-date by re-embedding during training. We answer this question affirmatively with the following highlights:

No Re-embedding Needed Training using target corrector networks matches the task performance of exhaustively re-embed all targets every 500 steps throughout training in both dense retrieval (Table 1) and retrieval augmented language models (Table 3). Target correctors achieve this without ever needing to re-embed targets during training, yielding significant computational savings (Fig. 2).

Best no re-embedding method Compared to frozen approaches, stale approaches, and Dynnibal without re-embedding, target corrector networks achieve over 10 point improvements in RLM tasks and 4 point improvements across multiple recall measures in retrieval.

Simpler and Less Computation Target correctors perform as well or better than Stochastic Negative Mining (SNM) (Reddi et al., 2019) despite SNM doing more re-embedding. Similarly, target corrector networks nearly match Dynnibal (Monath et al., 2023) when Dynnibal uses much more computation (Table 1). Dynnibal is a much more complicated and difficult to implement method.

5.1. Supervised Dense Retrieval

Setting & Metrics We evaluate training methods for supervised dense retrieval models. Each method is provided the same supervised data. All methods use a stale buffer of target representations and use this buffer to form the subset of targets, $S(\mathcal{Y})$, used in computing the truncated softmax. All methods use the same loss (cross-entropy) and optimize parameters of the dual-encoders using gradient descent. The methods differ in their maintenance of the buffer, and, as such, differ in their computational requirements of maintaining this buffer. We measure the computational requirements in terms of how many targets are re-embedding during training[1]. We measure re-embedding in terms of the number of targets encoded to indicate the computational expense (even if wall clock time is mitigated using a complicated asynchronous computation). Re-embedding every target even one additional time during training can be problematic if number of targets is large. Furthermore, the initial buffer, created using the initial parameters of the dual-encoder (e.g., a pre-trained language model) can be computed once and used for subsequent training jobs.

Data We evaluate on Natural Questions (Kwiatkowski et al., 2019) with over 21M targets (Wikipedia passages), about 60K training examples (question, passage pairs), and about 3K in dev/test, and MSMARCO (Bajaj et al., 2016) 8.8M targets (web passages), and 500K training examples.

Models We initialize the dual encoder models with two publicly available pre-trained language models, GTR (Ni et al., 2022), and T5 (Raffel et al., 2020). GTR is an encoder model initialized from T5 and further pre-trained for dense retrieval on a large collection of corpora of question/answer pairs. For MSMARCO, we only use T5 since it is included in GTR's training data. We use the base size models, $D = 768$, and train separate parameters for $f(x)$ and $g(y)$. For the target corrector, we use a two layer MLP. We use 8192 hidden units, a ReLU non-linearity, and a residual connection.

We compare the following approaches: **Target Corrector Networks** (this paper): At the first training step, we initialize the buffer with vector representations of every target. At every subsequent step, we use the target corrector network to produce a new representation of the targets, without running the target-encoder, simply by running our small MLP corrector on the stale representations. The stale buffer representations are never updated during training. **Stale**: We initialize the buffer of targets at the first step of training and do not update it throughout training. We experimented with both freezing the target encoder parameters $g(y)$ and allowing them to be updated despite the stale buffer. We found updating the parameters to be slightly better and report those results. **Exhaustive**: We exhaustively re-embed all of the targets in the buffer every 500 steps of training. **Stochastic Negative Mining** (SNM; Reddi et al. 2019): Instead of storing every target in the buffer, we store a subset of targets sampled uniformly at random. We re-sample and re-embed this buffer every 500 steps. We use a buffer size

[1] Our JAX (Bradbury et al., 2018) implementation run on Cloud TPUv3 re-embeds ~2184 targets per second on each core.

		Re-embed Num. (↓)	NQ Dev - Recall (↑) @1	@5	@10	@20	@100	NQ Test - Recall (↑) @1	@5	@10	@20	@100
GTR-base	In-batch	0	17.14	46.77	58.71	69.45	85.54	37.92	64.76	72.54	78.28	87.00
	Stale	0	33.11	62.04	70.31	78.13	89.32	46.76	68.64	75.21	80.66	87.48
	Dynnibal^{+}	0	28.73	59.66	70.08	78.14	90.18	44.40	67.53	74.93	80.22	87.23
	Corrector $_{(\ell_{\text{mse}})}$	0	34.98	65.03	74.01	80.77	90.82	49.61	70.72	77.04	82.33	88.28
	Corrector	0	**35.78**	**66.74**	**75.06**	**81.52**	**91.37**	**50.61**	**71.00**	**77.73**	**82.66**	**88.39**
	Dynnibal^{+}	42M	35.86	66.54	75.04	81.40	91.27	50.55	71.69	78.25	83.35	88.73
	SNM†	80M	32.03	64.01	73.72	81.37	91.47	49.14	69.89	77.12	82.19	87.95
	Exhaustive	1.68B	36.29	67.08	75.55	82.07	91.73	50.30	71.55	78.12	82.83	88.59
T5-base	In-batch	0	9.93	28.07	37.17	45.54	64.06	23.40	47.50	56.39	65.34	77.97
	Stale	0	16.79	36.85	44.82	51.79	67.35	27.65	50.19	59.28	66.98	78.95
	Dynnibal $^{+}$	0	17.42	39.65	48.75	57.36	73.03	29.72	53.99	63.38	70.61	80.94
	Corrector	0	**23.64**	**47.69**	**56.68**	**64.65**	**79.03**	**36.65**	**59.25**	**68.06**	**73.71**	**83.13**
	Dynnibal $^{+}$	42M	23.71	46.63	55.75	63.88	79.46	36.65	59.31	67.65	74.46	83.13
	Dynnibal $^{+}$	80M	24.76	47.69	56.82	64.90	80.15	36.90	59.97	68.23	74.54	83.35
	SNM †	80M	22.55	46.86	55.72	64.19	80.40	35.93	59.06	67.48	73.66	82.85
	Exhaustive	1.68B	24.70	48.21	57.18	65.39	79.94	37.34	60.42	68.70	74.76	83.41

Table 1. **Natural Questions** (Kwiatkowski et al., 2019). This paper; † (Reddi et al., 2019)' + (Monath et al., 2023). We measure the performance of dense retrieval models trained with different softmax approximations via hard negatives. We find that our proposed approach nearly matches the performance of exhaustively re-embedding all targets. Similarly, our approach requires significantly less re-embedding of targets than Dynnibal, which requires at least one exhaustive re-embedding to get competitive results. **Bold-face numbers** indicate best performance with zero re-embeddings performed at training time; underlined numbers indicate best performance using re-embedding at training time.

	Hidden Units	Steps/sec	R@1	R@20	R@100
Exhaustive	-	0.43	36.29	82.07	91.70
Corrector	8192	0.83	35.78	81.52	91.37
Corrector	4096	1.10	35.53	81.63	91.07
Corrector	2048	1.83	35.55	81.07	91.08

Table 2. **Steps-Per-Second Comparison.** We compare on NQ-dev the performance and speed of corrector networks and exhaustive re-encoding. Models are finetuned GTR-base.

of 1M targets. **Dynnibal** (Monath et al., 2023): This complicated approach maintains a buffer using a low-rank regression model as a part of tree index structure. The regression model is updated every 500 steps on a sub-sample of targets, unlike our approach which is trained jointly. Furthermore, to get good performance, Dynnibal performs costly full buffer re-embedding periodically throughout training. We needed to perform two such re-embeddings. Dynnibal with fewer refreshes does not perform as well.

In Table 1, our target corrector network approach greatly improves upon the stale approach, especially in Recall@1, 5, 10. We observe a nearly 5 point improvement at R@10 in the dev set and a 4 point improvement in R@1 on the test over the stale approach. Our approach nearly matches the performance of the computationally intensive exhaustive approach. Furthermore, we perform comparably to the more expensive SNM and Dynnibal methods. We perform better than Dynnibal for the same amount of re-embedding. While doubling the number of index refreshes may appear negligible, having to re-embed the buffer during training can be computationally burdensome, especially as the number of targets grows. Using a buffer created from the initial parameters of the dual-encoder as with our approach, allows the buffer to be constructed once ahead of time and re-used across both training and tasks. Dynnibal requires hand tuning to get the re-embedding schedule correct.

Table 1 also compares dual-encoder initialization. GTR is pre-trained for retrieval and hence achieves better results. T5 is not pre-trained for retrieval and requires more adaptation for the retrieval task. We observe that SNM struggles more to match the performance of Exhaustive with T5. Furthermore, Dynnibal requires more full index refreshes to get competitive results. Our method is able to achieve nearly as good results as the Exhaustive approach and Dynnibal (with re-embedding) despite never needing to re-embed.

We also report timing comparisons in terms of steps-per-second between corrector networks (of two sizes) and exhaustive re-encoding of the targets. These can be found in table 2. We can see that both small and large corrector networks lead large speed gains over exhaustive re-encoding with minimal performance gains. This indicates that corrector networks can have practical training time efficiency gains over exhaustive methods.

	Re-embed Num. (↓)	Retr.	NQ	TQA	HPQA
No Retr.	0	-	25.4	26.1	14.5
Frozen Retr.	0	GTR	48.4	55.1	28.0
Corrector	0	GTR	52.3	66.4	36.7
Exhaustive	1.1B	GTR	52.4	66.5	33.8
Frozen Retr.	0	T5	13.34	12.15	13.37
Corrector	0	T5	48.1	63.73	21.97
Exhaustive	1.1B	T5	48.3	66.03	25.45

Table 3. **Exact Match Accuracy, RLM Training**. We find that our target corrector approach can match the performance of the fully refreshed index while never re-embedding the targets across NQ, TriviaQA (TQA), and HotPotQA (HPQA).

See Appendix B.2 for additional results (MSMARCO, other ablations) and further discussion.

5.2. Retrieval Augmented Language Models

Setting & Metrics We evaluate the latent variable use case of training the retriever in a retrieval-augmented language model (RLM), as described in Section 3.1. We will compare approaches for training in terms of their re-embedding costs.

Datasets We evaluate on the three question answering datasets: TriviaQA (Joshi et al., 2017), NQOpen (Kwiatkowski et al., 2019), and HotPotQA (Yang et al., 2018). We use 256 token passages from a 2018 Wikipedia snapshot as the collection of targets, $\mathcal{Y}$, with 28M targets.

Models We initialize the retriever with GTR-base or T5-base and use T5-base as the reader in Fusion-In-Decoder (Izacard & Grave, 2020). We use 32 retrieved documents in all experiments. The target corrector is a two-layer MLP.

We compare the following approaches: **Target Corrector Network**: Target corrector is used to retrieve $S(\mathcal{Y})$ at training time. We embed the targets at the beginning of training and never update the buffer. **No Retrieval** (Roberts et al., 2020): The retriever is not used. The reader model is trained on the dataset and uses only its parameters to answer the questions. **Frozen Retrieval** (Izacard & Grave, 2020): Every target is embedded once at the beginning of training. Only the parameters of the reader model are trained (updating the retriever parameters did not improve performance). **Exhaustive**: Jointly training the retriever and reader, we exhaustively re-embed all 28M targets every 500 steps.

In Table 3, we report exact match accuracy on the held-out validation sets. Our proposed target corrector matches or nearly matches the performance of the exhaustive re-embedding approach without ever having to re-embed the buffer. This is a dramatic reduction in computational cost, as the exhaustive approach ends up embedding all 28M passages 40 times (1.1B re-embeddings). Target correctors greatly outperform the approaches that do not use retrieval (by more than 20 points) and the frozen retriever approach (by at least 4 points and by up to 10 points).

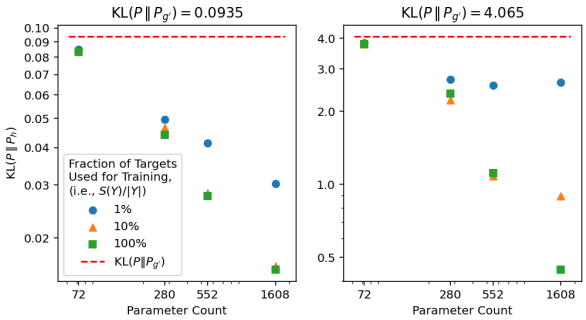

Figure 3. **Training Sample Size** The figure shows the trade-offs between the complexity of h (parameter count), the approximation error, $KL(P||P_h)$, and the fraction of samples used for training. The left hand side shows only somewhat stale representations. The right hand side shows significantly stale representations. Using a higher fraction of training samples is needed with more staleness.

5.3. Synthetic Experiments

In these experiments, we measure the ability of proposed corrector network to approximate categorical distributions parameterized by the softmax by training the corrector network, h, without training parameters of the dual-encoder.

Setting & Metrics We will measure the ability of proposed corrector network to approximate categorical distributions parameterized by the softmax. We do so by training the corrector network, h, in isolation, e.g., only training the parameters of the corrector network, Ψ, without training parameters of the dual-encoder for a particular task. We measure the quality of approximation using the KL-divergence between the true categorical distribution $P(y|x)$ (Equation 1) and the approximate distribution given by the corrector network $P_h(y|x)$ (Equation 4). We measure the complexity of the corrector network by its parameter count, $|\Psi|$. We measure staleness, i.e., the difficulty of correcting a set of stale representations, by the KL-divergence between the true categorical distribution $P(y|x)$ and the distribution $P_{g'}(y|x) \propto \exp(\beta\langle f(x), g'(y)\rangle)$.

Data Generation We directly generate vector representations corresponding to data points and targets. That is, rather than having a dual-encoder model provide the vector representation of a data point or target, we directly generate synthetic data corresponding to $f(x)$, $g(y)$, and $g'(y)$. We generate 4096 targets in $D = 8$ dimensions from a mixture of 20 Gaussians to represent $g'(y)$. To generate $g(y)$, we transform $g'(y)$ by feeding the points into randomly initialized MLPs with up to 2 hidden layers of size $D, 2D, 4D$ or $8D$, with RELU activation and residual connections. We vary the complexity of the MLP and variance of the initialization to create embeddings $g(y)$ to model a variety of settings of the extent of the staleness ($\mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y)$).

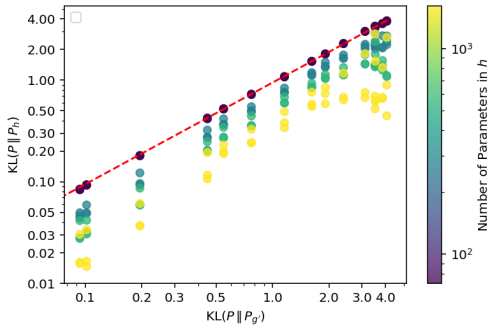

Figure 4. **Parameter Count** We plot the KL divergence using the stale embeddings ($\mathrm{KL}(P\|P_{g'})$, on the x-axis) against that of the trained correction models ($\mathrm{KL}(P\|P_h)$, on the y-axis). Parity is indicated by the dashed red line, demonstrating that the trained correction model is significantly better than using the stale embeddings. Increasing the parameter count of h is more important when the discrepancy between stale and current embeddings is higher, indicated by a larger improvement toward the right of this plot.

Corrector Network In these experiments, we vary the parameter count of the corrector network h and number of hidden layers, using between 0 and 2 hidden layers with hidden dimension of $D, 2D, 4D$, or $8D$. We use ReLU nonlinearity with residual connections. We optimize the parameters of the corrector network using Adam with learning rate 0.03, and stop when the loss has not improved for at least 100 epochs or we reach 1000 epochs of training.

Varying $|S(\mathcal{Y})|$, number of targets used for training In Figure 3, we explore trade-offs between the complexity in terms of the parameter count $|\Psi|$ of h (x-axis); the approximation error $\mathrm{KL}(P\|P_h)$ after applying the trained correction model (y-axis); and the fraction of samples used for training h. We report the complexity of the transformation from g' to g in terms of $\mathrm{KL}(P\|P_{g'})$ above each pane. Using a higher fraction of training samples is needed when there is more staleness. When the drift is more significant (right-hand pane), we observe that using increased parameters with a smaller fraction of samples does lead to overfitting. In this setting, it seems that sampling 10% of the targets is generally sufficient.

Varying Complexity of the Target Corrector Network In order to explore how the KL divergence of our approximation may change with respect to the staleness of the embeddings g', we train our embedding model to approximate the distributions P. In Figure 4, we explore how the KL divergence of our approximation may change with respect to the staleness of the embeddings g', We can obtain a significant reduction in KL divergence via the correction model (on the y-axis) across a wide variety of drifts (as measured by $\mathrm{KL}(P\|P_{g'})$). Increasing parameter count is always effective, but it yields greater benefit when approximating a distribution with greater divergence.

6. Related work

Energy-based Models Many similar ideas of training small parametric models to aid the training of other models has been widely studied in energy-based models, such as CoopNets (Xie et al., 2018), VERA (Grathwohl et al., 2021), and others (Grathwohl et al., 2020). In this setting models can be trained to skirt around intractable computations required in main-model training.

Amortized Inference There are many approaches that speed up sampling by fitting parametric models such as feed-forward neural networks (Marino et al., 2018; Naderiparizi et al., 2022).

Softmax Approximations Previous work has considered approximations to softmax via kernel methods (Blanc & Rendle, 2018; Rawat et al., 2019) when there are trainable parameters for every target (rather than an encoder). Sampling-based approaches are widely used as well (Vembu et al., 2009; Zaheer et al., 2017; Monath et al., 2023).

Adapters Adapter methods, which train small parametric components of larger networks (Houlsby et al., 2019) bear resemblance to our approach. However, our approach is distinct in that it operates only on the output layer of the neural models, not intermediate layers.

7. Conclusion

We present target corrector networks for approximating the softmax function during the training of dual encoder models and retrieval augmented language models. The target corrector networks learn to update a stale buffer of target representations. We investigate the generalization properties of the corrector models theoretically. We furthermore show empirically how our correct model approach can be used to train models (both supervised retrievers and retrieval augmented language models) matching the accuracy of models that use 4x-80x the computational budget during training.

Impact Statement

Our work proposes new more efficient ways of training of retrieval models. Retrieval models both in their own right and in combination with language models have wide and applicable uses. The techniques of this paper are about improving training efficiency. As such, better models could be produced faster, bringing to bear all the responsibilities of model creators in terms of understanding the successes, limitations, and biases of the model. Future work could consider the question of how different training strategies affect the way in which retrieval models have broad impact. Of particular interest to this paper could be the way in which staleness when computing the truncated softmax plays a role in such a study.

References

Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Blanc, G. and Rendle, S. Adaptive sampled softmax with kernel based sampling. In *International Conference on Machine Learning (ICML)*, 2018.

Bradbury, J., Frostig, R., Hawkins, P., Johnson, M. J., Leary, C., Maclaurin, D., Necula, G., Paszke, A., VanderPlas, J., Wanderman-Milne, S., and Zhang, Q. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/google/jax.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2019.

Devroye, L., Györfi, L., and Lugosi, G. *A probabilistic theory of pattern recognition*, volume 31. Springer Science & Business Media, 2013.

Dulac-Arnold, G., Evans, R., van Hasselt, H., Sunehag, P., Lillicrap, T., Hunt, J., Mann, T., Weber, T., Degris, T., and Coppin, B. Deep reinforcement learning in large discrete action spaces. *arXiv preprint arXiv:1512.07679*, 2015.

Gillick, D., Kulkarni, S., Lansing, L., Presta, A., Baldridge, J., Ie, E., and Garcia-Olano, D. Learning dense representations for entity retrieval. In *Conference on Computational Natural Language Learning (CoNLL)*, 2019.

Gottipati, S. K., Sattarov, B., Niu, S., Pathak, Y., Wei, H., Liu, S., Blackburn, S., Thomas, K., Coley, C., Tang, J., et al. Learning to navigate the synthetically accessible chemical space using reinforcement learning. In *International conference on machine learning*, pp. 3668–3679. PMLR, 2020.

Grathwohl, W., Wang, K.-C., Jacobsen, J.-H., Duvenaud, D., and Zemel, R. Learning the stein discrepancy for training and evaluating energy-based models without sampling. In *International Conference on Machine Learning*, pp. 3732–3747. PMLR, 2020.

Grathwohl, W., Kelly, J., Hashemi, M., Norouzi, M., Swersky, K., and Duvenaud, D. No mcmc for me: Amortized sampling for fast and stable training of energy-based models. *ICLR*, 2021.

Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M. Retrieval augmented language model pre-training. In *International Conference on Machine Learning (ICML)*, 2020.

Han, T., Nijkamp, E., Zhou, L., Pang, B., Zhu, S.-C., and Wu, Y. N. Joint training of variational auto-encoder and latent energy-based model. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 7978–7987, 2020.

Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, 2019.

Izacard, G. and Grave, E. Leveraging passage retrieval with generative models for open domain question answering, 2020. URL https://arxiv.org/abs/2007.0128.

Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. 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)*, pp. 1601–1611, 2017.

Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics (TACL)*, 2019.

Lindgren, E., Reddi, S. J., Guo, R., and Kumar, S. Efficient training of retrieval models using negative cache. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Logeswaran, L., Chang, M.-W., Lee, K., Toutanova, K., Devlin, J., and Lee, H. Zero-shot entity linking by reading entity descriptions. In *Association for Computational Linguistics (ACL)*, 2019.

Marino, J., Yue, Y., and Mandt, S. Iterative amortized inference. In *Proceedings of the 35th International Conference on Machine Learning*, pp. 3403–3412, 2018.

Mohri, M., Rostamizadeh, A., and Talwalkar, A. *Foundations of machine learning*. MIT press, 2018.

Monath, N., Zaheer, M., Allen, K., and McCallum, A. Improving dual-encoder training through dynamic indexes for negative mining. *AISTATS*, 2023.

Naderiparizi, S., Scibior, A., Munk, A., Ghadiri, M., Baydin, A. G., Gram-Hansen, B. J., De Witt, C. A. S., Zinkov, R., Torr, P., Rainforth, T., et al. Amortized rejection sampling in universal probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pp. 8392–8412. PMLR, 2022.

Ni, J., Qu, C., Lu, J., Dai, Z., Ábrego, G. H., Ma, J., Zhao, V. Y., Luan, Y., Hall, K. B., Chang, M.-W., et al. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*, 2021.

Ni, J., Qu, C., Lu, J., Dai, Z., Abrego, G. H., Ma, J., Zhao, V., Luan, Y., Hall, K., Chang, M.-W., et al. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9844–9855, 2022.

Qu, Y., Ding, Y., Liu, J., Liu, K., Ren, R., Zhao, W. X., Dong, D., Wu, H., and Wang, H. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2021.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Rawat, A. S., Chen, J., Yu, F., Suresh, A. T., and Kumar, S. Sampled softmax with random fourier features. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Rawat, A. S., Menon, A. K., Veit, A., Yu, F., Reddi, S. J., and Kumar, S. Doubly-stochastic mining for heterogeneous retrieval. *arXiv preprint arXiv:2004.10915*, 2020.

Reddi, S. J., Kale, S., Yu, F., Holtmann-Rice, D., Chen, J., and Kumar, S. Stochastic negative mining for learning with large output spaces. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

Roberts, A., Raffel, C., and Shazeer, N. How much knowledge can you pack into the parameters of a language model? *arXiv preprint arXiv:2002.08910*, 2020.

Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

Vembu, S., Gärtner, T., and Boley, M. Probabilistic structured predictors. In *Uncertainty in Artificial Intelligence (UAI)*, 2009.

Villani, C. et al. *Optimal transport: old and new*. Springer, 2008.

Wachsmuth, H., Syed, S., and Stein, B. Retrieval of the best counterargument without prior topic knowledge. In Gurevych, I. and Miyao, Y. (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, July 2018.

Xie, J., Lu, Y., Gao, R., and Wu, Y. N. Cooperative learning of energy-based model and latent variable model via mcmc teaching. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Xiong, L., Xiong, C., Li, Y., Tang, K.-F., Liu, J., Bennett, P. N., Ahmed, J., and Overwijk, A. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2020.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W., Salakhutdinov, R., and Manning, C. D. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018.

Yu, H.-F., Zhong, K., Zhang, J., Chang, W.-C., and Dhillon, I. S. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 2022.

Zaheer, M., Kottur, S., Ahmed, A., Moura, J., and Smola, A. Canopy fast sampling with cover trees. In *International Conference on Machine Learning (ICML)*, 2017.

A. Analysis: Proofs

Lemma 4.1. *Given a target encoder g and its stale approximation g', the gap between the true population risk and stale population risk is bounded in the following way:*

$$R_{\ell,\phi}(\mathcal{D}_Y) - R_{\ell,\phi}(\tilde{\mathcal{D}}_Y) \leq \mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y) \leq \|g - g'\|_1 \tag{13}$$

where $\mathcal{W}$ is the Wasserstein distance. Furthermore, if the approximation g' comes from the same neural model as g with parameters perturbed by u as in aforementioned stale approximation, we have: $\|g - g'\|_1 \leq L\|u\|$ with L as the Lipschitz constant.

Proof. We bound the gap between true population risk and stale population risk. Recall that $\mathcal{G}_{\ell,\mathcal{F}}$ is the induced function class: $\mathcal{G}_{\ell,\mathcal{F}} = \{y \mapsto \ell(f(y), g(y)) : f \in \mathcal{F}\}$. Now note that

$$\begin{aligned}
&R_{\ell,f}(\mathcal{D}_Y) - R_{\ell,f}(\tilde{\mathcal{D}}_Y) && (16)\\
&= \mathbb{E}_{Y\sim\mathcal{D}_Y}\left[\ell(f(Y), g(Y)\right] - \mathbb{E}_{Y'\sim\tilde{\mathcal{D}}_Y}\left[\ell(f(Y'), g(Y')\right]\\
&\leq \sup_{\lambda\in\mathcal{G}_{\ell,\mathcal{F}}}\Big(\mathbb{E}_{Y\sim\mathcal{D}_Y}\left[\lambda(X)\right] - \mathbb{E}_{Y'\sim\tilde{\mathcal{D}}_Y}\left[\lambda(Y')\right]\Big)\\
&\overset{(i)}{=} L\cdot\sup_{\lambda\in\mathcal{G}_{\ell,\mathcal{F}}}\left(\mathbb{E}_{Y\sim\mathcal{D}_Y}\left[\frac{\lambda(Y)}{L}\right] - \mathbb{E}_{Y'\sim\tilde{\mathcal{D}}_Y}\left[\frac{\lambda(Y')}{L}\right]\right)\\
&\overset{(ii)}{\leq} L\cdot\sup_{\lambda\in\mathrm{Lip}_1(\rho)}\Big(\mathbb{E}_{X\sim\mathcal{D}_Y}\left[\lambda(X)\right] - \mathbb{E}_{Y'\sim\tilde{\mathcal{D}}_Y}\left[\lambda(Y')\right]\Big),\\
&\overset{(iii)}{=} \mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y) && (17)\\
&\overset{(iv)}{\leq} \|\mathcal{D}_Y - \tilde{\mathcal{D}}_Y\|_{TV} && (18)\\
&\overset{(v)}{=} \frac{1}{2}\sum_{y\in\mathcal{Y}}|\mathsf{softmax}(g(y)) - \mathsf{softmax}(g'(y))| && (19)\\
&\overset{(vi)}{\leq} \frac{1}{2}\|g - g'\|_1 && (20)
\end{aligned}$$

where (i) follows by dividing and multiply by L; (ii) follows as, for any $\lambda \in \mathcal{G}^h_{\ell,\mathcal{F}}$, we have $\frac{\lambda}{L}$ to be 1-Lipschitz; (iii) follows from Kantorovich-Rubinstein duality (Villani et al., 2008); (iv) follows from Corollory 6.14 in Villani et al. (2008); (v) follows from definition; and (vi) follows from softmax Lipschtiz constant being 1. As g and g' are output from the same neural network but with parameters perturbed by u, then it follows that $\|g - g'\|_1 \leq L\|u\|$. □

Lemma 4.2. *Given a target encoder g, its stale approximation g', and a set of n-targets $\tilde{\mathcal{S}}_n = \{y_1, ..., y_n\}$ sampled from $\tilde{\mathcal{D}}_Y$,*

$$R_{\ell,\tilde{\phi}_n}(\tilde{\mathcal{D}}_Y) \leq R_{\ell,\tilde{\phi}_n}(\tilde{\mathcal{S}}_n) + \mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}}), \tag{14}$$

where $\mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}})$ is the Rademacher complexity of $\mathcal{G}_{\ell,\mathcal{F}}$.

Proof We need to connect the stale population risk to the empirical risk we are actually minimizing:

$$\begin{aligned}
R_{\ell,\tilde{f}_n}(\tilde{\mathcal{D}}_Y) &= \mathbb{E}_{\tilde{\mathcal{D}}_Y}[\ell(\tilde{f}_n(Y), g(Y))]\\
&\leq \mathbb{E}_{\tilde{\mathcal{S}}_n}[\ell(\tilde{f}_n(Y), g(Y))] + \sup_{f\in\mathcal{F}}\left|\mathbb{E}_{\tilde{\mathcal{D}}_Y}[\ell(f(Y), g(Y))] - \mathbb{E}_{\tilde{\mathcal{S}}_n}[\ell(f(Y), g(Y))]\right|\\
&\overset{(i)}{=} \mathbb{E}_{\tilde{\mathcal{S}}_n}[\ell(\tilde{f}_n(Y), h(X))] + \sup_{g\in\mathcal{G}_{\ell,\mathcal{F}}}\left|\mathbb{E}_{\tilde{\mathcal{D}}_Y}[g(Y)] - \mathbb{E}_{\tilde{\mathcal{S}}_n}[g(Y)]\right|\\
&\overset{(ii)}{\leq} \mathbb{E}_{\tilde{\mathcal{S}}_n}[\ell(\tilde{f}_n(Y), h(X))] + \mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}})\\
&= R_{\ell,\tilde{f}_n}(\tilde{\mathcal{S}}_n) + \mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}}), && (21)
\end{aligned}$$

where inequality (i) follows from the definition of $\mathcal{G}_{\ell,\mathcal{F}}$ and (ii) from the standard symmetrization argument (Devroye et al., 2013; Mohri et al., 2018) for Radamacher complexity. □

Theorem 4.3. *For a target encoder, g, its stale approximation, g', and the Rademacher complexity $\tilde{\mathfrak{R}}_n(\mathcal{G}_{\ell,\mathcal{F}^{g'}})$, the true population risk $R_{\ell,\phi}(\mathcal{D}_Y)$ is bounded by the following with probability at least $1-\delta$:*

$$\begin{aligned} R_{\ell,\phi}(\mathcal{D}_Y) &\leq T_1 + T_2 + T_3 \\ T_1 &= R_{\ell,\tilde{\phi}_n}(\tilde{\mathcal{S}}_n) \\ T_2 &= \mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y) \leq L\|u\| \\ T_3 &= \tilde{\mathfrak{R}}_n(\mathcal{G}_{\ell,\mathcal{F}^{g'}}) + \mathcal{O}\Big(\sqrt{\frac{\log(1/\delta)}{n}}\Big) \end{aligned} \tag{15}$$

Proof. As mentioned in the text $\mathcal{F}$ might be too large function class and we would like to utilize the restricted function class $\mathcal{F}^{g'}$. The previous derivation would go through using this restricted class and we will obtain the Rademacher complexity of $\mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}^{g'}})$ instead. To compare the two Rademacher complexity, observe that

$$\begin{aligned} \mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}^{g'}}) &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{\lambda\in\mathcal{G}_{\ell,\mathcal{F}^{g'}}}\sum_i \sigma_i\lambda(y_i)\right| \\ &\overset{(i)}{=} \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{h\in\mathcal{H}}\sum_i \sigma_i\ell(h\circ g'(y), g(y))\right| \\ &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{f\in\mathcal{F}^{g'}}\sum_i \sigma_i\ell(f(y), g(y))\right| \\ &\overset{(ii)}{\leq} \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{f\in\mathcal{F}}\sum_i \sigma_i\ell(f(y), g(y))\right| \\ &= \frac{1}{n}\mathbb{E}_{\boldsymbol{\sigma}}\left|\sup_{\lambda\in\mathcal{G}_{\ell,\mathcal{F}}}\sum_i \sigma_i\lambda(y_i)\right| \\ &= \mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}}), \end{aligned} \tag{22}$$

where (i) follows from definition of $\mathcal{G}_{\ell,\mathcal{F}^{g'}}$ and $\mathcal{F}^{g'}$; and (ii) holds because $\mathcal{F}^{g'} \subset \mathcal{F}$.

Now, the standard concentration results for empirical Rademacher complexity implies that, with probability at least $1-\delta$, we have the following.

$$\mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}^{g'}}) \leq \mathbb{E}_{\tilde{\mathcal{S}}_n\sim\tilde{\mathcal{D}}_Y^{\otimes n}}\left[\mathfrak{R}_{\tilde{\mathcal{S}}_n}(\mathcal{G}_{\ell,\mathcal{F}^{g'}})\right] + \mathcal{O}\Big(\sqrt{\frac{\log(1/\delta)}{n}}\Big) \tag{23}$$

$$= \tilde{\mathfrak{R}}_n(\mathcal{G}_{\ell,\mathcal{F}^{g'}}) + \mathcal{O}\Big(\sqrt{\frac{\log(1/\delta)}{n}}\Big). \tag{24}$$

Combining results from Eq. 16, 21, and 23, we obtain that with probability at least $1-\delta$,

$$R_{\ell,f}(\mathcal{D}_Y) \leq R_{\ell,\tilde{f}_n}(\tilde{\mathcal{S}}_n) + \underbrace{\mathcal{W}(\mathcal{D}_Y, \tilde{\mathcal{D}}_Y)}_{\leq L\|u\|} + \tilde{\mathfrak{R}}_n(\mathcal{G}_{\ell,\mathcal{F}^{g'}}) + \mathcal{O}\Big(\sqrt{\frac{\log(1/\delta)}{n}}\Big) \tag{25}$$

□

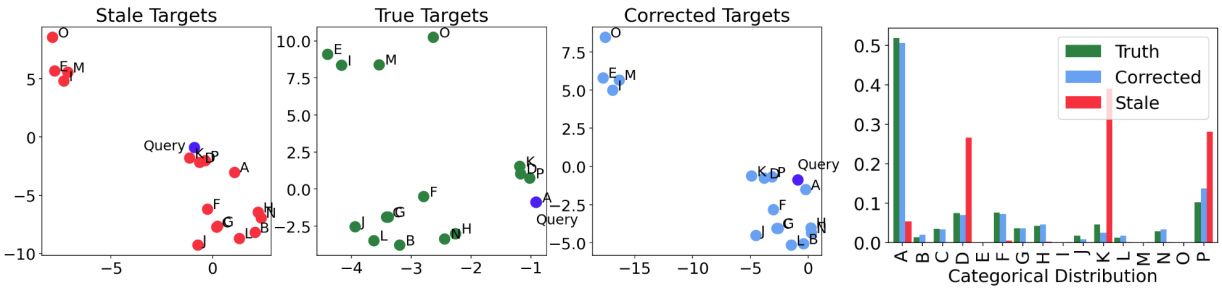

Figure 5. A toy experiment where stale and true targets are distributed around the unit circle, and the corrected targets based on the learned approximation are also depicted. The associated distribution over targets ($\beta = 20$) based on the point identified.

Performance	Num. Re-Embed	R@1	R@5	R@10	R@100
Stale	0	10.11	27.70	36.33	63.69
SNM	20M	18.18	43.48	54.68	82.18
Dynnibal	8M	18.23	43.15	54.56	82.24
Target Corrector	0	17.07	40.78	51.56	79.29
Exhaustive	352M	18.18	44.97	55.58	83.69

Table 4. **MSMARCO, T5 Initialization.** In this experiment, we measure performance of methods on MSMARCO. With fewer targets than Natural Questions, the gap between stochastic negative mining and the refreshed index is reduced.

B. Experiments

B.1. Experimental Details

Training Details We train all models, the dual-encoders and the corrector model, jointly using Adam (Kingma & Ba, 2014). We implement the training procedure using stop-gradients so that the corrector model loss only changes the corrector model parameters and dual-encoder loss the dual-encoder ones. We form the subset of targets for the truncated softmax, $S(\mathcal{Y})_x$, using the top-64 closest targets to the given query according to a particular training procedure's buffer and 64 targets chosen uniformly at random. We use a minibatch size of 128 examples and share the truncated softmax targets across all examples in the minibatch mb , e.g., $\bigcup_{x \in mb} S(\mathcal{Y})_x$. We use 40K steps for retrieval training and 20K steps for RLM training. We combine the task losses and corrector network loss together. We experimented with a weight parameter applied to the corrector network. We use a weight value of 10.0.

B.2. Additional Dense Retrieval Results

In Table 4, we report performance on MSMarco using T5-base as the encoder. Here, with fewer targets, Stochastic Negative Mining provides a better approximation as a larger fraction of targets is re-encoded. Our method is still able to nearly match the performance of the exhaustive approach. We are able to achieve such results without having to re-embed the buffer.

Using Accelerator Memory to Store the Buffer In these experiments, we store the buffer of targets on the accelerator, making implementation of our approach training extremely easy. However, it could be the case that not all targets can fit into a buffer on accelerator memory. In such settings, our approach could still be used in the following ways: (1) subsample targets randomly (perhaps changing the subset periodically) to fit on device memory akin to a combination of our corrector approach stochastic negative mining, which would require no re-encoding of targets, or (2) use our approach to re-rank stale representations initially retrieved from CPU memory.

Comparisons to 2-Round Training Several recent works such as (Qu et al., 2021) which addresses difficulties of training dense retrieval models proposes to train in 2 stages. First all targets are encoded (using random or pre-trained model). Then the model is trained for one half of the desired iterations. Then the new model's parameters are used to re-encode the targets a single time. Then the model is trained for the remaining steps using these re-encoded targets. We compare this approch with corrector networks in Table 5. We see that when using GTR-base, the performance for all methods is quite similar (with corrector networks and exhaustive re-encoding slightly outperforming). When T5-base is used though, we find the

Method	Base	R@1	R@5	R@10	R@20	R@100
Two Round	T5	29.50	53.40	62.49	70.64	80.94
Corrector	T5	36.65	59.25	68.06	73.71	83.13
Exhaustive	T5	37.34	60.42	68.70	74.76	83.41
Two Round	GTR	49.06	70.06	76.76	81.17	87.95
Corrector	GTR	49.61	70.72	77.04	82.33	88.28
Exhaustive	GTR	50.30	71.55	78.12	82.83	88.59

Table 5. **Comparison with 2-round training.** We compare the performance of corrector networks, exhaustive re-encoding, and 2-round training. Results are presented on the NQ Test set.

Method	Base	R@1	R@100
With Uniform	T5	24.70	79.94
Without Uniform	T5	24.87	79.82
With Uniform	GTR	36.29	91.73
Without Uniform	GTR	36.53	91.70

Table 6. **Uniform Negatives Comparison.** We compare the performance of exhaustive re-encoding with and without the addition of uniform (in-batch) negatives. Results are presented on the NQ Dev set.

performance of corrector networks and exhaustive re-encoding to notably out-perform the 2-step procedure. We attribute this to GTR being a better initialization for the model. In this case we would expect its parameters (and therefore its target embeddings) to change less from pre-training to fine-tuning, meaning that there is less embedding drift and therefore less bias when using the 2-step procedure.

Comparisons with and without uniform negatives In our main experiments (as stated in Appendix B.1) we train with hard negatives *and* uniform negatives. Initial experiments showed that adding uniform negatives lead to improved performance in some settings. We provide some additional results ablating this choice using exhaustive re-encoding. These can be found in Table 6. We can see that this choice provides negligible improvement on the reported benchmarks (although we believe its worth trying in other settings).

B.3. Retrieve and Read

Note that in this setting we do not share the subset of targets $S(\mathcal{Y})$ across the examples in the batch, nor do we use targets sampled uniformly at random.

The versions of the retrieve-and read datasets are:

- TriviaQA: https://www.tensorflow.org/datasets/catalog/trivia_qa#trivia_qaunfilterednocontext
- NQOpen https://www.tensorflow.org/datasets/catalog/natural_questions_open
- HotPotQA https://www.tensorflow.org/datasets/community_catalog/huggingface/hotpot_qa

Performance	P@10	P@20	P@100
$g'(\cdot)$	67.57	67.73	57.57
$h \circ g'(\cdot)$	76.87	73.32	53.55

Table 7. **Approximating Large Models with Small Models** On the dataset Arguana (Wachsmuth et al., 2018), we use our method to warp the embedding space of GTR Small so that it is better aligned with GTR Large. Note that here we present nearest neighbor precision, i.e., the overlap in the top-K neighbors from the large model at 10, 20, and 100. We use 32 samples for each query to train the correction model.

B.4. Beyond Stale Representations: Approximating Large Models with Small Models

In this experiment, we focus on sampling in isolation. We sample a batch of input points and we measure the ability of our method to approximate one dual encoder model with another. In particular, we study a case where we approximate a large dual encoder with a small model. We approximate the GTR large model (Ni et al., 2021) (e.g., $g(\cdot)$) with the GTR small model(e.g., $g'(\cdot)$). In Table 7, we report nearest neighbor precision, i.e., measuring the overlap in the top-K neighbors from the large model's neighbors at 10, 20, and 100 on the dataset Arguana (Wachsmuth et al., 2018) and other BEIR benchmark datasets (Thakur et al., 2021). We use 32 samples for each query to train the correction model. We find that overlap amongst smaller K seems to be better aligned using our method.