Improving Dual-Encoder Training through Dynamic Indexes for Negative Mining

Nicholas Monath
Google Research

Manzil Zaheer
Google DeepMind

Kelsey Allen
Google DeepMind

Andrew McCallum
Google Research

Abstract

Dual encoder models are ubiquitous in modern classification and retrieval. Crucial for training such dual encoders is an accurate estimation of gradients from the partition function of the softmax over the large output space; this requires finding negative targets that contribute most significantly ("hard negatives"). Since dual encoder model parameters change during training, the use of traditional static nearest neighbor indexes can be sub-optimal. These static indexes (1) periodically require expensive re-building of the index, which in turn requires (2) expensive re-encoding of all targets using updated model parameters. This paper addresses both of these challenges. First, we introduce an algorithm that uses a tree structure to approximate the softmax with provable bounds and that dynamically maintains the tree. Second, we approximate the effect of a gradient update on target encodings with an efficient Nyström low-rank approximation. In our empirical study on datasets with over twenty million targets, our approach cuts error by half in relation to oracle brute-force negative mining. Furthermore, our method surpasses prior state-of-the-art while using 150x less accelerator memory.

1 INTRODUCTION

Dual encoder models map input queries and output targets to a common vector space in which inner products of query and target vectors yield an accurate similarity function. They are a highly effective, widely deployed solution for classification and retrieval tasks such as passage retrieval (Karpukhin et al., 2020), question answering (Qu et al., 2021), recommendation systems (Wu et al., 2020), entity linking (Gillick et al., 2019), and fine-grained classification (Xiong et al., 2022). These tasks are characterized as

having a large number of targets, often on the order of millions to billions. Dual encoders achieve scalability to this large number of targets in two ways: weight sharing among targets through a parametric encoder and an efficient inner product-based scoring function. The encoder models are often parameterized as deep neural networks, e.g., transformers (Vaswani et al., 2017; Devlin et al., 2019), and trained with the cross entropy loss between the model's softmax distribution and query's true labeled target(s).

Training dual encoder models *efficiently* and *effectively* poses two key challenges:

Computationally intensive loss function. Computing the gradient of the softmax partition function becomes computationally costly when the number of possible targets is large (Bengio and Senécal, 2008; Daumé III et al., 2017; Lindgren et al., 2021, inter alia), necessitating approximation. The common approach approximates the large sum in the partition function gradient by sampling relatively few of its largest terms originating from "hard" negative targets, which are often found using an efficient nearest-neighbor index (Guu et al., 2020; Agarwal et al., 2022). However, the approximation introduces bias in gradient estimation affecting learning and resulting accuracy (Rawat et al., 2019; Ajalloeian and Stich, 2020).

Moving embeddings. The embedded representations of both queries and targets continuously change during training as the underlying encoder parameters are updated. Since re-embedding all targets after each step of training is computationally infeasible, prior work uses 'stale' representations for negative mining, i.e., the vector representation from the encoder parameters from t steps ago. Re-encoding and re-indexing even at moderately-sized intervals remains an expensive operation (Izacard et al., 2022).

In this paper, we present a new dual-encoder training algorithm, which we call DYNNIBAL, that addresses both of the above challenges, and we provide both theoretical and empirical analysis. To elaborate:

Efficient gradient bias reduction. The expensive term in the exact gradient computation is the evaluation of an expectation with respect to the entire softmax distribution. To approximate this expectation, we design a Metropolis-

Hastings sampler that uses a tree-structured hierarchical clustering of the targets to provide an efficient proposal distribution in §3. We then relate bias reduction to the granularity of the clustering. We also present an efficient method for dynamically updating this tree structure in response to encoding parameter gradients.

Efficient re-embedding. Instead of frequently re-running the updated (typically fairly large and expensive) target encoder model to produce up-to-date embeddings of targets, we propose rather to approximate (with effective end-task performance) the effect of a gradient update on the target embeddings with an efficient Nyström low-rank approximation (generally using several orders of magnitude less time and memory) in §4.

Theoretical analysis. We study the running time of our algorithm under mild assumptions such as Lipschitz encoder functions and expansion rate, showing per-step cost is a function of the number of clusters in our approximation, considerably smaller than the number of targets itself (Prop. 2, Prop. 3). We also present a bound on the bias of our softmax estimator, ensuring convergence with stochastic gradient descent (Prop. 5).

Empirical performance. In §5, we evaluate our algorithm on two passage retrieval task datasets, *Natural Questions (NQ)* (Kwiatkowski et al., 2019) and *MSMARCO* (Bajaj et al., 2016). On NQ, which has over twenty million targets, we find that our approach cuts error by half in relation to a practically-infeasible oracle brute-force negative mining. Previous state-of-the-art methods can incrementally increase accuracy by increasing memory usage at training time; yet we find that our method still surpasses previous state-of-the-art when using 150x less accelerator memory.

2 BACKGROUND

Given a data point $x \in \mathcal{X}$ (e.g., a query), we are tasked with predicting a target $y \in \mathcal{Y}$ (e.g., a passage answering the question). In our experiments, the number of targets is large, such as tens of millions. We assume that the targets y are featurized. We use *encoder* models to represent both the points and targets. These encoders, which map a point or target's features to a fixed dimensional embedding, are often large parametric pre-trained models (such as transformers (Vaswani et al., 2017; Devlin et al., 2019)). We denote the encoder model for points as $f_\theta(x) \in \mathbb{R}^d$ and for targets as $f_\phi(y) \in \mathbb{R}^d$. The softmax distribution is:

$$P(y|x) = \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{Z \triangleq \sum_{y' \in \mathcal{Y}} \exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)}, \quad (1)$$

where β denotes the inverse temperature hyperparameter.

We train the parameters of the encoder models $\Theta = \{\theta, \phi\}$, given labeled training data pairs $(x_1, y_1), \ldots, (x_N, y_N)$. Our training objective is the cross entropy loss, which for a given training pair is defined as:

$$\mathcal{L}(x_i, y_i) = -\beta\langle f_\theta(x_i), f_\phi(y_i)\rangle + \log Z. \quad (2)$$

As mentioned by Rawat et al. (2019), most methods use first order optimization, so we consider the gradient wrt Θ:

$$\nabla_\Theta \mathcal{L}(x_i, y_i) = -\nabla_\Theta \beta\langle f_\theta(x_i), f_\phi(y_i)\rangle + \nabla_\Theta \log Z$$
$$\nabla_\Theta \log Z = \mathbb{E}_{y \sim P(y|x_i)} \nabla_\Theta \beta\langle f_\theta(x_i), f_\phi(y)\rangle.$$

Training with cross-entropy loss is computationally challenging because computing the partition function, Z, or its gradient requires $|\mathcal{Y}|$ inner-products. Furthermore, it requires $|\mathcal{Y}|$ encoding calls to the target encoding model. This latter challenge is unique to training dual encoders and is not a challenge when targets have their own free parameters (e.g., Daumé III et al. (2017); Sun et al. (2019)).

Many works replace the expensive full expectation computation with a Monte Carlo estimate from a small constant number of samples obtained in different ways (Henderson et al., 2017; Reddi et al., 2019; Rawat et al., 2019; Karpukhin et al., 2020; Lindgren et al., 2021). In our work, we design a novel proposal distribution from which it is efficient to sample, and which has bounded gradient bias, to ensure faster convergence of SGD (Ajalloeian and Stich, 2020). We refer to our approximate loss as $\hat{\mathcal{L}}$.

Our work will use the same, mild assumptions of many previous works (Rawat et al., 2019; Lindgren et al., 2021).

Assumption 1. (Lipschitz Encoders) *The dual encoders are L-Lipschitz in parameters Θ, that is $||f_\theta(x) - f_{\theta'}(x)|| \leq L||\Theta - \Theta'||$ and $||f_\phi(y) - f_{\phi'}(y)|| \leq L||\Theta - \Theta'||$ for all $x \in \mathcal{X}$, $y \in \mathcal{Y}$.*

Assumption 2. (Bounded Gradients) *We assume that the logits have bounded gradients, $||\nabla\langle f_\theta(x), f_\phi(y)\rangle|| < M$.*

Assumption 3. (Unit Norm) *Dual-encoders produce unit normed vector embeddings[1], $\forall y \in \mathcal{Y}$, $||f_\phi(y)|| = 1$.*

3 EFFICIENT AND ACCURATE SAMPLES FROM THE SOFTMAX DISTRIBUTION

In this section, we present our approach for maintaining a dynamic tree-structured clustering that supports efficient and provably accurate sampling from the softmax distribution. In the next section (§4), we will show a novel use of low-rank regression-based approximations to obviate the need for using the encoder to produce updated embeddings and describe the complete training algorithm. Proofs for all statements are relegated to the Supplement. Given the properties of the proposed method, we refer to our approach as DYNNIBAL, in reference to **Dy**namic **N**earest **N**eighbor **I**ndex for **B**ias-reduced **A**pproximate **L**oss (Figure 1).

---

[1]We note that it is common practice to unit norm the representations from dual encoders, e.g., (Gillick et al., 2019; Rawat et al., 2020; Lindgren et al., 2021)

Nicholas Monath, Manzil Zaheer, Kelsey Allen, Andrew McCallum

3.1 Accurate Samples from Softmax Distribution

We would like to accurately sample from the softmax distribution, without having to compute the computationally intensive the partition function, Z. We consider familiar methods: rejection sampling and Metropolis-Hastings.

To apply rejection sampling, we approximate the unnormalized softmax probability for each target y with an approximation $\hat{y}$ such that:

$$e^{-\epsilon_r} \leqslant \frac{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)} \leqslant e^{\epsilon_r}. \quad (3)$$

Then, if we sample from:

$$y \sim \frac{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}{\sum_{y'} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle)} \quad (4)$$

and accept with probability

$$e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}, \quad (5)$$

we will sample from the softmax, akin to past work on rejection sampling for mixture models (Zaheer et al., 2017).

Similarly, we can use Metropolis-Hastings to produce a sample from the true softmax distribution $P(y|x)$ by iteratively sampling (and accepting/rejecting) a state change from a proposal distribution, denoted Q. The approximation error in terms of the total variation of the distribution given by Metropolis-Hastings, Q_{MH}, compared to the true softmax distribution P using a s-length chain can be bounded by (Mengersen and Tweedie, 1996; Cai, 2000; Bachem et al., 2016):

$$||P - Q_{\mathsf{MH}}||_{\mathsf{TV}} \leqslant \exp\left(-\frac{s-1}{\gamma}\right) \quad \gamma = \max_{y\in\mathcal{Y}} \frac{P(y|x)}{Q(y|x)}. \quad (6)$$

This means that $s > 1 + \gamma \log \frac{1}{\epsilon}$ gives $||P - Q_{\mathsf{MH}}||_{\mathsf{TV}} \leqslant \epsilon$.

We achieve high quality samples if the proposal distribution Q is 'close' to the true softmax in terms of the ratio P/Q. From high quality samples, we will see that bias is minimized to aid convergence of SGD.

3.2 Clustering-based Approximations

The main idea of our approach is to build a clustering of the targets that quantizes the true softmax distribution P by assigning each target to a cluster such that targets in the same cluster have the same probability. Ideally, these clustering-based approximations would be efficient to construct/maintain, and would minimize approximation error.

As an introduction, consider a flat-clustering based approach. We denote the clustering of targets as $\mathcal{C}$. We denote the cluster assignment of the target y as $y^{(\mathcal{C})}$. Each cluster in the clustering $\mathcal{C} \in \mathcal{C}$, $\mathcal{C} \subseteq \mathcal{Y}$ is associated with a representative. We overload notation and use $\mathcal{C}$ and $y^{(\mathcal{C})}$ to
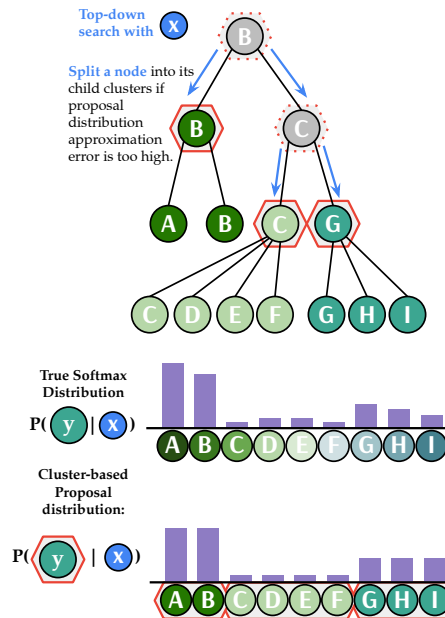

Figure 1: **Proposed Approach: DyNNIBAL**. Our approach searches the tree structured index for a clustering of the labels. This clustering is then used to provide an approximate softmax distribution used as a proposal distribution for Metropolis-Hastings. The tree structure supports efficient updates as parameters of the dual encoders change.

refer to both (1) the set of targets (when used in context of clustering) as well as (2) the features of the cluster's representative (when used as input to a dual encoder model).

We define the distribution Q, by replacing each target with a representative for its cluster.

$$Q(y|x; \mathcal{C}) = \frac{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C})})\rangle)}{\hat{Z} \triangleq \sum_{\mathcal{C}\in\mathcal{C}} |\mathcal{C}| \exp(\beta\langle f_\theta(x), f_\phi(\mathcal{C})\rangle)}, \quad (7)$$

The complexity of sampling from this distribution is a function of the number of clusters, or rather, we only need to measure similarity between the datapoint x and each of the cluster representatives, e.g., $\mathcal{O}(|\mathcal{C}|)$ inner products to compute the unnormalized probabilities and partition function. We investigate how to discover a clustering of targets which provides efficient sampling while bounding the error of Metropolis-Hastings and increasing the probability of acceptance for rejection sampling.

3.3 Hierarchical Clustering Structures

Our method for discovering such a clustering is to use hierarchical clustering, in particular SG Trees (Zaheer et al., 2019), an efficient variant of cover trees (Beygelzimer et al., 2006). These efficient hierarchical approaches will allow us to bound the approximation quality and discover adaptive clustering of targets for each point.

Definition 1. (Hierarchical Clustering) *A hierarchical clustering $\mathcal{T}$ is a set of nested clusterings. For $\mathcal{C}_{\mathsf{par}}$, 'child'*

clusters are $\mathsf{ch}(\mathcal{C}_{\mathsf{par}})$ *s.t.* $\forall \mathcal{C}_{\mathsf{kid}} \in \mathsf{ch}(\mathcal{C}_{\mathsf{par}}), \mathcal{C}_{\mathsf{kid}} \subsetneq \mathcal{C}_{\mathsf{par}}$ *and* $\nexists \mathcal{C}'$ *s.t. for some* $\mathcal{C}_{\mathsf{kid}} \in \mathsf{ch}(\mathcal{C}_{\mathsf{par}}), \mathcal{C}_{\mathsf{kid}} \subset \mathcal{C}' \subset \mathcal{C}_{\mathsf{par}}$.

Cover trees, originally proposed as a nearest neighbor index, are highly scalable to hundreds of millions of targets (Zaheer et al., 2017) and enjoy theoretical guarantees.

Definition 2. (**Cover Tree** (Beygelzimer et al., 2006)) *A cover tree with base* b *is a* level-wise *structure. Levels are clusterings,* $\mathcal{C}_{(\ell)}$. *The set of cluster representatives in a level is* $Y_{(\ell)}$. *A cover tree maintains the invariants:*

1. **Nesting**. *The cluster representatives at a parent level are a subset of the child level, i.e.,* $Y_{(\ell)} \subseteq Y_{(\ell-1)}$.

2. **Covering**. *For all* $y \in Y_{(\ell-1)}$, *there exists a parent node* $y_{\mathsf{par}} \in Y_{(\ell)}$ *such that* $||f_\phi(y) - f_\phi(y_{\mathsf{par}})|| \leqslant b^\ell$

3. **Separation**. *All distinct nodes in a given* $y, y' \in Y_{(\ell)}$, *satisfy* $||f_\phi(y) - f_\phi(y')|| \geqslant b^\ell$.

Closely related are SG Trees (Zaheer et al., 2019):

Definition 3. (**Stable Greedy (SG) Tree** (Zaheer et al., 2019)) *An SG tree is a cover tree, with separation defined to only apply to siblings rather than nodes in the same level:*

3. **Separation**. *All distinct siblings,* $y, y' \in \mathsf{ch}(y_{\mathsf{par}})$, *satisfy* $||f_\phi(y) - f_\phi(y')|| \geqslant b^\ell$.

Because SG trees are considerably more efficient to construct (Zaheer et al., 2019), they are the focus of our work. However, where possible, we will also describe how cover trees could be used in our methods.

The cover tree and SG tree data structures are not new to this paper. Their use to provide an approximation to the softmax is novel and is the contribution of this section.

We make standard assumptions about the representations (Beygelzimer et al., 2006; Zaheer et al., 2019, inter alia).

Definition 4. *The expansion constant* α *for the encoded targets* $Y = \{f_\phi(y) : y \in \mathcal{Y}\}$ *is the smallest* $\alpha \geqslant 2$ *such that* $|B(p, 2r)| \leqslant \alpha |B(p, r)|$ *for all* $p \in Y$ *and* $r \geqslant 0$ *where* $B(p, r)$ *denotes a ball of radius* r *around* p.

First, notice how a particular level of the tree structure can serve as a clustering used in the approximate distribution. Each cluster in the given level has bounded radius over its descendants (cluster members). By bounding the approximation error of the unnormalized and normalized probabilities for a selected clustering, we can determine the quality of samples using Metropolis-Hastings as well as the acceptance probability for rejection sampling.

Proposition 1. *Under Assumption 3, given the clustering at level* ℓ, $\mathcal{C}_{(\ell)}$, *approximating* y *with the cluster representative,* $y^{(\mathcal{C}_{(\ell)})}$, *satisfies the following with* $\epsilon_r = \beta \cdot b^\ell$:

$$e^{-\epsilon_r} \leqslant \frac{\exp(\beta \langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})}) \rangle)}{\exp(\beta \langle f_\theta(x), f_\phi(y) \rangle)} \leqslant e^{\epsilon_r}. \quad (8)$$

Similarly, to achieve a given bound on the total variation for Metropolis-Hastings, we would like to control γ,

the ratio of the true softmax to our proposal distribution, $\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q(y|x; \mathcal{C}_{(\ell)})} = \gamma$, in terms of the level ℓ selected.

Proposition 2. *Given Assumption 3, to achieve a maximum ratio of true softmax to proposal distribution equal to* γ *i.e.,* $\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q(y|x; \mathcal{C}_{(\ell)})} = \gamma$, *we need the clustering at level* ℓ, *where:* $\ell \triangleq \max\{\ell \in \mathbb{Z} : b^\ell \leqslant \frac{1}{2\beta} \log \gamma\}$.

Remark 1. *Observe the relationship between* γ *and* ℓ. *Descending one more level of the tree to level* $\ell - 1$, *reduces the ratio from* γ *(selecting level* ℓ*) to* $\gamma^{\frac{1}{b}}$.

The aforementioned results describe how to achieve a given approximation error by selecting a level of the tree structure. Now, let's consider methods which adaptively use the hierarchical structure to produce a sample for a given datapoint. These approaches will start with a coarse clustering (e.g., some level ℓ satisfying a minimal requirement on the aforementioned sources of error).

First, let's consider a theoretically motivated rejection sampling approach based on the rejection sampling methods for mixture models proposed by Zaheer et al. (2017). The approach works by iteratively selecting a finer-grained set of cluster representatives to sample from while still being a valid rejection sampler for the softmax distribution. The sampling is modified such that if a given cluster is accepted, we return its representative. We start with the clusters at level ℓ. We perform one step of rejection sampling. If we accept, we return the cluster representative itself. Otherwise, we descend to that cluster's children in level $\ell - 1$ and repeat the following procedure until the leaves of the tree. We sample among the children of the node and one specially defined *restart* option, $\circlearrowleft$. If the *restart* option is sampled, we begin the algorithm again at level ℓ. If we sample a child other the nested self-child and we accept the child's cluster, we return its representative with a given probability. If we sample the nested-child or if we do not accept the sampled child, we descend the tree and consider the children of the sampled node. To be a valid rejection sampler, we maintain running normalizers as shown in Algorithm 1.

Proposition 3. *Algorithm 1 produces samples from the softmax* $P(y|x)$ *in time* $\mathcal{O}(|\mathcal{C}_\ell| + \alpha^4 e^{\beta b^{\ell+2}})$ *for cover trees and* $\mathcal{O}(|\mathcal{C}_\ell| + \alpha^3 e^{\beta b^{\ell+2}})$ *for SG trees.*

In Proposition 3, the first term of the cost corresponds to computation and second term to quality of the proposal. If we pick ℓ to be too close to the root, then computation is cheaper
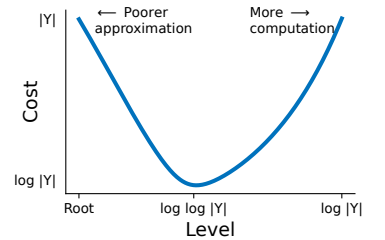
as cluster size $|\mathcal{C}_\ell|$ will be small, but quality of the proposal will be worse. As we descend down the down the tree for picking ℓ the quality of proposal will keep improving but so will computation cost as $|\mathcal{C}_\ell|$ grows. There is a good trade-off point in between as illustrated in the figure.

Algorithm 1 REJECTIONSAMPLING

1: **Input:** $\mathcal{T}$: tree, x: point, ℓ: initial level.
2: **Output:** A sample from $P(y|x)$
3: Define $\blacktriangle_{\mathcal{C}} \leftarrow \frac{|\mathcal{C}|}{|\mathcal{Y}|}$ for all tree nodes $\mathcal{C} \in \mathcal{T}$.
4: $Z_\ell \leftarrow e^{b^\ell} \sum_{\mathcal{C} \in \mathcal{C}_i} \blacktriangle_{\mathcal{C}} \exp(\beta \langle f_\theta(x), f_\phi(\mathcal{C}) \rangle)$
5: Define $\delta_{\ell,\mathcal{C}} \leftarrow Z_\ell^{-1} e^{b^\ell} \blacktriangle_{\mathcal{C}} \exp(\beta \langle f_\theta(x), f_\phi(\mathcal{C}) \rangle)$
6: Sample $\mathcal{C}_\ell$ from $\mathcal{C}_\ell$ proportional to $\delta_{\ell,\mathcal{C}_\ell}$
7: Accept & **return** the representative of $\mathcal{C}_\ell$ with prob:
$$\pi_\ell \leftarrow Z_\ell^{-1} \delta_{i,\mathcal{C}_i}^{-1} |\mathcal{Y}|^{-1} \exp(\beta \langle f_\theta(x), f_\phi(\mathcal{C}_i) \rangle)$$
8: **for** k **from** $\ell - 1$ **down to** $-\infty$ **do**
9: $\quad Z_k \leftarrow \delta_{k+1}(1 - \pi_{k+1})$
10: $\quad \delta_{k,\mathcal{C}_k} \leftarrow Z_k^{-1} e^{b^k} \blacktriangle_{\mathcal{C}_k} \exp(\beta \langle f_\theta(x), f_\phi(\mathcal{C}_k) \rangle)$
11: $\quad \delta_{k,\bigcirc} \leftarrow 1 - \sum_{\mathcal{C} \in \mathsf{ch}(\mathcal{C}_{k+1})} \delta_{k,\mathcal{C}}$
12: $\quad$ Sample $\mathcal{C}_k$ from $\mathsf{ch}(\mathcal{C}_{k+1}) \bigcup \{\bigcirc\}$ prop. to $\delta_{k,\mathcal{C}_k}$.
13: $\quad$ **if** $\mathcal{C}_k = \bigcirc$ **then**
14: $\quad\quad$ REJECTIONSAMPLING$(\mathcal{T}, x, \ell)$
15: $\quad$ **else if** the representative of $\mathcal{C}_k$ & $\mathcal{C}_{k+1}$ differ **then**
16: $\quad\quad$ Accept & **return** the rep. of $\mathcal{C}_k$ with prob π_k:
$$\pi_k \leftarrow Z_k^{-1} \delta_{k,\mathcal{C}_k}^{-1} |\mathcal{Y}|^{-1} \exp(\beta \langle f_\theta(x), f_\phi(\mathcal{C}_k) \rangle)$$

Next, we consider a more practical and simple adaptive extension for Metropolis-Hastings. Intuitively, we want to reduce approximation error for high probability targets. Our approach splits a cluster if the radius is at least b^m and if it *may* contain a target y, such that $||f_\theta(x) - f_\phi(y)|| < b^m$. This follows the intuition that we would like to emphasize the closest targets to a point. In other words, we descend the tree from level ℓ to level m splitting clusters which might contain a target that is within b^m of $f_\theta(x)$ (Algorithm 2).

Proposition 4. *Let $\mathcal{C}$ be the output of Algorithm 2, then* $\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q(y|x,\mathcal{C})} \leqslant \gamma$ *under Assumption 3.*

Remark 2. *While in the worst case, we select all clusters in level m, e.g., $\mathcal{O}(|\mathcal{C}_{(m)}|)$ clusters, in practice we expect this to be much less and can limit to a specified number by either limiting the size of the frontier or output partition.*

To put our results in perspective, consider a simple alternative, uniform negative sampling. Let Q_{unif} be a uniform proposal distribution, for which $\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q_{\mathsf{unif}}(y|x)} \leqslant |\mathcal{Y}|$.

Remark 3. (Uniform Negatives vs DyNNIBAL) *Consider the case where we have fixed compute budget for the chain length. For a uniform distribution, the total variation is bounded by $\exp\left(-\mathcal{O}\left(\frac{s}{|\mathcal{Y}|}\right)\right)$. Since our each sample is slightly more expensive, we can only afford to have a chain of length $\frac{s}{|\mathcal{C}_{(\ell)}|}$ for any selected clustering $\mathcal{C}_{(\ell)}$. But even this reduced length chain will yield a much better total variation bound by Proposition 1. In particular, if we pick a level just $\log \log |\mathcal{Y}|$ below the root, which is not very deep, then we obtain the total variation bound as $\exp\left(-\mathcal{O}\left(\frac{s}{|\log \mathcal{Y}|}\right)\right)$. Notice that this is marked improvement because of the logarithmic term in the denominator.*

Algorithm 2 FINDCLUSTERING

1: **Input:** $\mathcal{T}$: tree, x: point, γ, m: allowed error
2: **Output:** A clustering $\mathcal{C}$
3: $\ell \leftarrow \max\{\ell \in \mathbb{Z} \ : \ b^\ell \leqslant \frac{2}{\beta} \log \gamma^{\frac{1}{2}}\}$
4: $\mathcal{F}_\ell \leftarrow \mathcal{C}_\ell$ ▷ Initialize frontier to be the ℓth level of $\mathcal{T}$.
5: $\mathcal{C} \leftarrow \{\}$ ▷ The output clustering
6: **for** k **from** ℓ **down to** m **do**
7: $\quad \mathcal{F} \leftarrow \{\mathsf{ch}(F) : F \in \mathcal{F}_k\}$
8: $\quad \mathcal{F}_{k-1} \leftarrow \{\}$
9: $\quad$ **for** F in $\mathcal{F}$ **do**
10: $\quad\quad$ **if** $||f_\theta(x) - f_\phi(F)|| > b^k + b^m$ **then**
11: $\quad\quad\quad \mathcal{C} \leftarrow \mathcal{C} \cup \{F\}$
12: $\quad\quad$ **else**
13: $\quad\quad\quad \mathcal{F}_{k-1} \leftarrow \{F\}$
14: **return** $\mathcal{C} \cup \mathcal{F}_m$

3.4 Gradient Bias of Our Estimator

To ensure convergence in gradient descent, we need our estimator to have bounded gradient bias (Ajalloeian and Stich, 2020). We are interested in the bias of the gradient estimate: $||\mathbb{E}[\nabla_\Theta \hat{\mathcal{L}}] - \nabla_\Theta \mathcal{L}||$, where the expectation is over the Metropolis-Hasting samples.

Proposition 5. *Let P be the true softmax and Q_{MH} be the Metropolis-Hastings approximation to the softmax. Under Assumption 2, we have $||\mathbb{E}[\nabla_\Theta \hat{\mathcal{L}}] - \nabla_\Theta \mathcal{L}|| \leqslant 2\epsilon \beta M$, where $||P - Q_{\mathsf{MH}}||_{\mathsf{TV}} \leqslant \epsilon$.*

3.5 Dynamic Maintenance of the Tree Structure

During training, the parameters of the dual encoder models, f_θ, f_ϕ are updated. As a result the tree structure properties may no longer be upheld. In this section, we analyze how representations could change under standard assumptions about the data. We then describe an algorithm for maintaining an SG tree and a simple approximation in practice.

Finding the part of the SG tree that no longer maintains its invariants after a parameter update depends on how the distance between a pair of targets changes after w steps of gradient descent. Let the learning rate be η. The dual encoder parameters are updated at step t as $\Theta_t \leftarrow \Theta_{t-1} - \eta \nabla_\Theta \hat{\mathcal{L}}(x_t, y_t)$. We can bound the pairwise change:

Proposition 6. *Under Assumptions 1,2,3, let ϕ_t and ϕ_{t+w} refer to encoder parameters after w more steps of gradient descent with learning rate η.*

$$||f_{\phi_t}(y) - f_{\phi_t}(y')||_2 - ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||_2 \leqslant 4w\eta\beta LM. \quad (9)$$

We can detect whether, for a given pair of tree nodes with representatives y and y', if the covering property with respect to level ℓ will be maintained after the gradient update:

$$||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||_2 + 4w\eta\beta LM \leqslant b^\ell, \quad (10)$$

and similarly for separation:

$$||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||_2 - 4w\eta\beta LM \geqslant b^{\ell-1}. \quad (11)$$

Algorithm 3 UPDATESGTREE

1: **Input:** $\mathcal{C}_\ell$: a subtree root at level ℓ, $\mathcal{C}_r$: an ancestor node covering its descendants or $\varnothing$ to indicate we are rebuilding at the top level, $4w\eta\beta LM$: bound on change
2: **if** $b^\ell \leqslant 4w\eta\beta LM$ **then**
3:     Rebuild the subtree $\mathcal{C}_\ell$ under $\mathcal{C}_r$.
4: **else**
5:     MAXD$(\mathcal{C}_\ell)$ is the max dist btw $\mathcal{C}_\ell$ to a descendant.
6:     MIND$(\mathcal{C}_\ell)$ is the min dist btw children of $\mathcal{C}_\ell$.
7:     cov $\leftarrow$ MAXD$(\mathcal{C}_\ell) + 4w\eta\beta LM \leqslant b^\ell$
8:     sep $\leftarrow$ MIND$(\mathcal{C}_\ell) - 4w\eta\beta LM \geqslant b^{\ell-1}$
9:     **if** cov and sep **then**
10:         **for** $\mathcal{C}_{\ell-1} \in$ ch$(\mathcal{C}_\ell)$ **do**
11:             UPDATESGTREE$(\mathcal{C}_{\ell-1}, \mathcal{C}_\ell, 4w\eta\beta LM)$
12:     **else**
13:         Rebuild the subtree $\mathcal{C}_\ell$ under $\mathcal{C}_r$.

This leads to a simple algorithm for detecting which parts of the tree structure need to be re-arranged. We can maintain for each node, the distance to its farthest descendant, denoted MAXD$(\mathcal{C})$, to detect if covering is maintained. Similarly, we can maintain for each node, the distance between its closest pair of children nodes (SG tree), denoted MIND$(\mathcal{C})$, to detect if separation is maintained. Notice however that even if an ancestor node maintains the properties, a descendant may still violate them.

An algorithm for updating the SG tree is: delete and rebuild the smallest subtrees such that levels above the subtree root maintain covering and separation (checking MAXD and MIND)). We notice that for SG Trees since the separation property is only maintained between siblings and not all nodes in a given level, when we rebuild the structure we can re-attach the rebuild subtree with the same root. This is described in Algorithm 3. We observe that the selected bound $4w\eta\beta LM$ is equivalent to picking a level at which we rebuild, which can be easier in practice. Rebuilding subtrees can be empirically very efficient. The rebuilding can be done independently in parallel. Each subtree contains relatively far fewer targets than the tree as a whole.

4 EFFICIENT RE-ENCODING

We address further computational bottlenecks. First, the running time (and space) is a function of the dimensionality of dual encoder embeddings. This dimensionality can typically be quite large ($d = 768$ (Devlin et al., 2019)). Second, we previously implicitly described the representations of the targets stored in the tree, as $f_\phi(\mathcal{C})$ for some cluster representative $\mathcal{C}$. We do not re-apply the encoder every time we compare a given data point's embedding to a cluster representative. Instead, we use a cached version of the the target embedding. However, re-running the dual encoder to re-encode and update this cache (say every w steps of gradient descent) would be very time-consuming and require use of hardware accelerators (GPU/TPU).

We present approaches for addressing each of these com-

putational burdens. The Nyström method is used to reduce dimensionality. Then we use these low-dimensional representations in a low-rank regression model that approximates the re-running of the encoder model to produce the newest encoded representations (no accelerator required).

4.1 Reducing Dimensionality

The Nyström method factorizes a pairwise similarity matrix by representing each row and column (e.g., targets or datapoints) as a d' dimensional vector (Williams and Seeger, 2000; Kumar et al., 2012; Gittens and Mahoney, 2013, inter alia). Each dimension of a vector can be thought of as the (scaled) pairwise similarity between the representations of the row/column and a *landmark* representative associated with the particular dimension. Let $\mathbf{K} \in \mathbb{R}^{n \times n}$ be the pairwise similarity matrix, $\mathbf{S} \in \{0, 1\}^{n \times d'}, \forall j \in [d'] \sum_{i \in n} \mathbf{S}_{ij} = 1$ is an indicator matrix corresponding to the sampled landmarks, $n = |\mathcal{Y}| + |\mathcal{X}|$. The Nyström approximation is then, $\mathbf{KS} \left(\mathbf{S}^T \mathbf{KS} \right)^{-1} \mathbf{S}^T \mathbf{K}$. It is important to note that we do not actually *explicitly* instantiate $\mathbf{K}$. We sample our landmarks, compute the pairwise similarity between points/targets and the landmarks to compute $\mathbf{KS}$ and the landmarks themselves $(\mathbf{S}^T \mathbf{KS})^{-1}$. The low-dimensional representation of a target is the corresponding row in $\mathbf{KS} \left(\mathbf{S}^T \mathbf{KS} \right)^{-1}$ or $\mathbf{S}^T \mathbf{K}$. We can approximate all of the pairwise distance computations needed with an inner product of two d' dimensional vectors, where $d' \ll d$.

4.2 Regression-based Approximation of Re-Encoding

After w steps of gradient descent, our model parameters Θ_t have been updated to be Θ_{t+w}. We would like to approximate the re-encoding of targets $f_{\phi_{t+w}}(y)$ without having to run the encoder model. Let Y_t be the set of cached target embeddings after t steps, where $\vec{y}$ represents the cached vector for the target y. We want to build Y_{t+w} and we will do so by training a regression model, $\mathcal{R}$ to map each $\vec{y} \in Y_t$ to $f_{\phi_{t+w}}(y)$. We build s' training data pairs of the form, $(\vec{y_1}, f_{\phi_{t+w}}(y_1)), \ldots (\vec{y_{s'}}, f_{\phi_{t+w}}(y_{s'}))$. We then fit $\mathcal{R}$ using kernel ridge regression for which we use the above Nyström approximation to the kernel matrix.

To update the cached target representations, we apply the regression model $\mathcal{R}$ to every cached target in Y_t, creating the new cache of target representations Y_{t+w}. This provides an extreme speedup in re-encoding time. Instead of $\mathcal{O}(|\mathcal{Y}|)$ calls to the encoder, we have $\mathcal{O}(s')$ encoder calls where s' is the number of training examples for $\mathcal{R}$ and $s' \ll |\mathcal{Y}|$ along with one application of $\mathcal{R}$ to each target, effectively a multiply of an $d' \times d'$ matrix.

Further details can be found in Appendix B. The methodological techniques of Nyström and low-rank regression are not new, of course; our contribution is the use of these techniques to facilitate efficient re-encoding, which is a costly and time consuming step in dual encoder training.

Nicholas Monath, Manzil Zaheer, Kelsey Allen, Andrew McCallum

4.3 Complete Training Algorithm

In Algorithm 4, we summarize DYNNIBAL. Initially, we are given a set of targets $\mathcal{Y}$. We select (randomly) landmarks to apply Nyström. We encode and reduce the dimensionality of the targets. We construct an SG tree $\mathcal{T}$ over these targets. For a given training example x_t with label y_t, we use Algorithm 2 (with inputs of $\mathcal{T}$, the datapoint x and a given γ allowable error) to find a clustering of the labels $\mathcal{C}$. From this clustering, we can define the proposal distribution $Q(y|x, \mathcal{C})$ (Equation 7). Then, we use this proposal distribution to run a given number of Metropolis-Hastings steps to provide samples from the softmax distribution. The resulting samples are used to compute a loss for the given example, $\hat{\mathcal{L}}(x, y^\star)$, and the result in a gradient step for the dual-encoder parameters. After every w steps, we update the tree using Algorithm 3. We describe the algorithm as SGD for simplicity; a minibatch version is used in practice.

Algorithm 4 DYNNIBAL Training Algorithm

1: **Input:** $\mathcal{Y}$: Targets, $\mathcal{X}_{\text{train}} = \{(x_1, y_1), \ldots, \}$: Training, γ, m: allowed error, f_θ, f_ϕ: encoders, k: num samples, η: learning rate, U: update upper-bound
2: INITIALIZETARGETENCODING() ▷ Algorithm 6
3: Construct $\mathcal{T}$ ▷ Algorithm from Zaheer et al. (2019)
4: **for** t from 0 to NumSteps **do**
5:     Sample (x_t, y_t) from $\mathcal{X}_{\text{train}}$
6:     $\mathcal{C} \leftarrow$ FINDCLUSTERING$(\mathcal{T}, x_t, \gamma, m)$ ▷ Alg. 2
7:     $Q(y|x; \mathcal{C}) = \frac{1}{\hat{Z}} \exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C})})\rangle)$ ▷ Eq. 7
8:     Sample $N_s = \{y_{s_i} : i \in [k]\}$ via $y_{s_i} Q_{\text{MH}}, i \in [k]$.
9:     $\hat{Z} \leftarrow \exp\beta\langle f_\theta(x_t), f_\phi(y_t)\rangle + \sum_{i\in[k]} \exp\beta\langle f_\theta(x_t), f_\phi(y_{s_i})\rangle$
10:     $\hat{\mathcal{L}} \leftarrow -\beta\langle f_\theta(x_t), f_\phi(y_t)\rangle + \log(\hat{Z})$.
11:     $\theta \leftarrow \theta - \eta\nabla_\Theta\hat{\mathcal{L}}, \quad \phi \leftarrow \phi - \eta\nabla_\phi\hat{\mathcal{L}}$
12:     **if** t mod $w = 0$ **then**
13:         APPROXIMATEREENCODE() ▷ Algorithm 5
14:         $\mathcal{T} \leftarrow$ UPDATESGTREE$(\mathcal{T}, \varnothing, U)$ ▷ Algorithm 3
15: **return** f_θ, f_ϕ

---

5 EXPERIMENTS

We compare our proposed approach, DYNNIBAL, to multiple state-of-the-art methods for training dual encoders. We evaluate on two retrieval datasets and also the entity linking dataset, Zeshel (Logeswaran et al., 2019), in §D.

Natural Questions (NQ) (Kwiatkowski et al., 2019) is a dataset for passage retrieval. The data points are natural language questions. The targets are passages from Wikipedia. There are over 21 million targets and about 60K training examples. We report results using the string-match-based recall evaluation that is used by previous work (Karpukhin et al., 2020; Lindgren et al., 2021).

MSMARCO (Bajaj et al., 2016) contains 8.8 million targets and 500K training examples. Data points are natural language questions and targets are passages extracted from web documents. The task is to provide the correct passage

(target) for a given question (datapoint). Following previous work, we report the mean reciprocal rank.

We compare the following methods with dual encoders as transformer (Vaswani et al., 2017) initialized from pre-trained RoBERTa base (Liu et al., 2019). See details in §C.

In-batch Negatives. We approximate the softmax distribution by only using the positive target labels from within the batch as in previous work (e.g., Henderson et al. (2017))

Uniform Negatives. Sample targets uniformly at random from the collection of targets and use this to approximate the softmax distribution (e.g., Karpukhin et al. (2020)).

Stochastic Negative Mining (Reddi et al., 2019). A large number of targets is sampled uniformly at random. These targets are stored with their stale representations on the accelerator device. From this large number of targets, we approximate the softmax distribution with k sampled negatives. We periodically, e.g. every 100 or 500 steps refresh and change the negatives stored on the accelerator device. The performance of this method depends on how many targets are stored on the accelerator device. In most settings, memory becomes a bottleneck before the computational burden of computing pairwise similarities on the accelerator. We report performance in terms of this memory bottleneck, relative to the overall dataset size.

Negative Cache (Lindgren et al., 2021). A recent approach that keeps track of a collection of targets in a cache on accelerator similar to Stochastic Negative Mining. However, rather than randomly refreshing, negatives are added and removed the targets in a streaming manner, FIFO or LRU. We report results directly from the published paper.

Oracle Exhaustive Brute-Force. We exhaustively compute all logits and exhaustively find the top-k closest targets for training. Including this method provides insights into upper-bound empirical results, but it is impractical in practice since it is extremely expensive in computation and accelerator memory, and thus also financial cost. While we obtain results from this "oracle" method on the datasets above, running on meaningfully larger data would not have been possible even if computation/budget were no object.

DYNNIBAL. This paper's proposed cluster and approximate re-encoding approach. We note that the only portion of our training method sitting on accelerator memory is the low-rank regression-based approximate re-encoding model (§4.2), the landmark points, and regression training data. Our use of lower dimensional (64 or 128) Nyström embeddings along with scalable and relatively lightweight index structures stored in cheap CPU memory would allow our approach to scale to billions of targets, and further scale to many billions by leveraging the tree structure for additional targeted truncations and cluster-based approximations.

Empirically, we find that the Metropolis-Hasting-based sampling outperforms rejection sampling. We find using

Performance	Mem. %	R@1	R@5	R@10	R@20	R@100
In-batch Negatives	0.0%	0.356	0.613	0.695	0.757	0.843
Uniform Negatives	0.0%	0.386	0.644	0.723	0.775	0.848
DyNNIBAL (This Paper)	0.3%	0.485	0.695	0.754	0.801	0.862
Stochastic Negative Mining	1.0%	0.444	0.672	0.739	0.785	0.855
Stochastic Negative Mining	3%	0.461	0.689	0.750	0.794	0.860
Negative Cache	6.3%	-	-	-	0.784	0.856
Stochastic Negative Mining	10%	0.468	0.690	0.758	0.798	0.862
Exhaustive Brute Force	100%	0.500	0.700	0.765	0.804	0.866

Table 1: **Natural Questions**. We report the performance on the test set using the answer string retrieval recall (Karpukhin et al., 2020). Methods are listed in ascending order of required accelerator memory (as % of Exhaustive Brute Force memory). Scaling to many targets requires an approach that does not need accelerator memory, since storing such targets on the accelerator will become a limiting bottleneck in terms of memory. DyNNIBAL uses lower dimensional representations in CPU memory, an efficient tree update, and approximate re-encoding during training. DyNNIBAL performs significantly better than the practical approaches with $\leqslant 1.0\%$ memory. Error with respect to the Exhaustive Brute Force method is cut by half in terms of R@1 with respect to all methods other than the impractical 10% memory Stochastic Negative Mining. In more detail, DyNNIBAL (0.485 R@1) cuts error to 1.5 points compared to Exhaustive Brute Force (0.500), compared to Stochastic Negative Mining 3% (0.461) cuts error to 3.9 points. Compared to Stochastic Negative Mining 10% (0.468) which cuts error to 3.2, DyNNIBAL cuts error by over 2x more. In terms of R@5, DyNNIBAL also observes a strong 2 point gain over the low-memory variant of Stochastic Negative Mining (1.0% Mem.)

MRR	Mem. %	@1	@10	@100
In-batch Negatives	0	0.140	0.242	0.254
Uniform Negatives	0	0.196	0.305	0.316
Negative Cache	0.06%	-	0.310	-
Negative Cache	0.24%	-	0.315	-
DyNNIBAL	0.76%	0.223	0.334	0.345
Negative Cache	0.96%	-	0.323	-
Stochastic Neg.	1.0%	0.200	0.309	0.320
Stochastic Neg.	3.0%	0.216	0.331	0.342
Negative Cache	3.8%	-	0.322	-
Negative Cache	15.15%	-	0.331	-
Exhaustive	100%	0.228	0.345	0.356

Table 2: **MSMarco Results**. DyNNIBAL again outperforms competing methods that use similar low percentage memory requirements. In particular, DyNNIBAL outperforms 1% memory Stochastic Negative Mining in terms of MRR@1 by more than 2 points. Furthermore, we see that even when Negative Cache uses 15.15% memory, a 150x increase compared to our approach, DyNNIBAL still produces a higher MRR@10 result.

lower temperature and approximating the sampling procedure to ensure that "harder" negatives are selected is beneficial to end task performance. Rather than considering the entire partition of targets, $\mathcal{C}$, returned by Algorithm 2, we consider the top-k closest clusters. We set $m = -11$ (MS-MARCO) and $m = -14$ (NQ) as the deepest level of clusters and restrict the size of frontier in Algorithm 2 to be 100. We run chains of length 2. Finally, as another empirical approximation, we select the top scoring targets from all the chains. We discuss these choices further in §C.

5.1 Empirical Results

We report the performance of each method in terms of each dataset's given evaluation metric. Alongside the performance, we report the accelerator (GPU/TPU) memory requirement as a percentage with respect to the Oracle Exhaustive Brute-Force approach (storing all targets as full dimensional embeddings). As noted when describing Stochastic Negative Mining, we find that the main bottleneck for our competitors is the memory required to store target embeddings on the accelerators (along with the transformer encoders), not the computation of logits. Thus of crucial importance are DyNNIBAL's advantages over baselines with respect to reduced need for accelerator memory.

Table 1 shows the results on the test set for NQ. We observe that DyNNIBAL outperforms In-batch Negatives, Uniform Negatives, and Stochastic Negative Mining using 1.0% memory in all recall metrics. In terms of recall at 1, DyNNIBAL cuts the performance gap with respect to exhaustive brute force by more than half compared to all methods except 10% memory Stochastic Negative Mining. Even using 10% memory with Stochastic Negative Mining compared to 0.03% for DyNNIBAL leaves large performance gaps in terms of recall at 1. Stochastic Negative Mining achieves 0.468 versus DyNNIBAL's 0.485.

Table 2 reports performance for all methods on MS-MARCO in terms of mean reciprocal rank. Following past work (Lindgren et al., 2021), we evaluate on the development set. We find that performance on MSMARCO follows a similar trend as NQ in that DyNNIBAL outperforms all of the other low-memory approaches. In fact, even using 150x more memory Negative Cache still does not perform as well as DyNNIBAL. DyNNIBAL sees a multiple point improvement in MRR@1 compared to competing methods.

We consider the time needed to maintain the SG tree structure during training. We find on MSMARCO that, given a collection of updated target embeddings, our dynamic method (§3.5) is about 4x faster than the traditional approaches that rebuild the entire structure. Furthermore, we find that our regression-based approximate re-encoding is about 8x faster than running the encoder model on MSMARCO, and requires much less accelerator expense. In terms of training steps-per-second efficiency between index maintenance, our approach does involve more computation: it is about four times slower than Stochastic Negative Mining's simple logit-proportional sampling on a small subset of targets; however, some of that is balanced by faster index updates, and we expect that additional engineering design could significantly speed up our method; in any case, running Stochastic Negative Mining for more training steps did not increase its accuracy.

6   RELATED WORK

Per-Target Free Parameters. A large body of work trains classifiers with a massive number of targets. In much of this work, the targets are directly parameterized with a d dimensional vector, rather an embedding produced by an encoder (Bengio and Senécal, 2008; Choromanska and Langford, 2015; Jernite et al., 2017; Daumé III et al., 2017; Sun et al., 2019; Yu et al., 2022). Most closely related to our work are methods that adaptively re-arrange tree-structured classifiers (Sun et al., 2019; Jasinska-Kobus et al., 2021, inter alia). Blanc and Rendle (2018); Rawat et al. (2019) use kernel-based approximations of the softmax, which gives strong theoretical guarantees. Future work could consider how to extend these methods to the dual-encoder setting.

Partition Functions & Probabilistic Models. MCMC methods are widely used for posterior inference (Neal, 1993, 2000; Chang and Fisher III, 2013; Zaheer et al., 2016, 2017, inter alia). Unlike our setting, these methods are concerned with finding a high quality estimate of the distribution as the end task. In our setting, we need an estimate for every training point in each step of training. Most similar (and the inspiration for our approach) is the Canopy-sampler (Zaheer et al., 2017), uses a cover tree to derive an efficient rejection sampler/Metropolis-Hastings algorithm for exponential family mixture models. Unlike that work, we consider approximation of the softmax, moving embedded representations during training, and the relationship between the approximation quality and the gradient bias of our estimator. Vembu et al. (2009) use MCMC-based approximations of distributions with large output spaces, but does not use the clustering-based approximations presented here nor does it consider the dual-encoder setting.

Reparameterization, discrete distributions. OS* sampling, the Gumbel-Max trick and Perturb-and-MAP, are alternative methods for sampling from distributions such as the softmax (Dymetman et al., 2012; Tucker et al., 2017; Maddison et al., 2016; Paulus et al., 2020; Jang et al., 2016; Huijben et al., 2022). Future work could consider combining such reparameterization methods and our approach.

Nearest Neighbor Search, Clustering, Dynamic Structures. Cover Trees (Beygelzimer et al., 2006) and SG Trees (Zaheer et al., 2019) were originally used as nearest neighbor indexes. There are many tree and DAG-based index structures that support addition and deletion of items such as Navigating Nets (Krauthgamer and Lee, 2004), HNSW (Malkov et al., 2014), and NN-Descent (Dong et al., 2011). Also, closely related is work on maintaining dynamic hash-based indexes (Jain et al., 2008; Zhang et al., 2020). Apart from nearest neighbor index structures, scalable methods for hierarchical clustering organize large datasets into tree structures (Bateni et al., 2017; Moseley et al., 2019; Dhulipala et al.). Other work builds these clusterings in an incremental or dynamic setting (Liberty et al., 2016; Choromanska and Monteleoni, 2012; Kobren et al., 2017; Menon et al., 2019). Maintaining structures in a dynamic setting is studied in graph algorithms (e.g., minimum spanning tree). Efficient and parallelizable dynamic graph algorithms exist (Sleator and Tarjan, 1981; Holm et al., 2001; Tseng et al., 2019; Dhulipala et al., 2020, inter alia).

Cover Trees. New algorithms and extensions of cover trees have been developed (Curtin et al., 2013, 2015; Izbicki and Shelton, 2015; Zaheer et al., 2019; Elkin and Kurlin, 2021; Gu et al., 2022, inter alia). Cover trees are also widely used in other settings such as k-means clustering (Curtin, 2017) and Gaussian processes (Terenin et al., 2022).

Task Specific Related Work. Learned models for passage retrieval and entity linking are extensively studied (Wu et al., 2019; Karpukhin et al., 2020; Bhowmik et al., 2021; Qu et al., 2021; Thakur et al., 2021; Ren et al., 2021; Ni et al., 2021; FitzGerald et al., 2021; Gao and Callan, 2021; Dai et al., 2022; Izacard et al., 2022, inter alia). Most similar to our work is ANCE (Xiong et al., 2020), which uses a nearest neighbor index for a contrastive objective. Other work includes alternative architectures to dual encoders (Khattab and Zaharia, 2020; Qian et al., 2022; Santhanam et al., 2021) and learning efficient hashing-based representations (Yamada et al., 2021).

7   CONCLUSION

We present DYNNIBAL, a dynamic tree structure for clustering-based approximations of the softmax distribution. Our algorithm efficiently gives provably accurate samples for training dual-encoders with cross-entropy loss. Empirically, DYNNIBAL outperforms state-of-the-art on datasets with over twenty million targets, reducing error by half compared to an exhaustive oracle. We find that our dynamic maintenance of the tree structure can be 8x faster than exhaustive re-indexing. Furthermore, our approach outperforms state-of-the-art while using 150x less accelerator memory.

References

Dhruv Agarwal, Rico Angell, Nicholas Monath, and Andrew McCallum. Entity linking via explicit mention-mention coreference modeling. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2022.

Ahmad Ajalloeian and Sebastian U Stich. On the convergence of sgd with biased gradients. *arXiv preprint arXiv:2008.00051*, 2020.

Olivier Bachem, Mario Lucic, Hamed Hassani, and Andreas Krause. Fast and provably good seedings for k-means. *Advances in neural information processing systems (NeurIPS)*, 2016.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*, 2016.

Mohammadhossein Bateni, Soheil Behnezhad, Mahsa Derakhshan, MohammadTaghi Hajiaghayi, Raimondas Kiveris, Silvio Lattanzi, and Vahab Mirrokni. Affinity clustering: Hierarchical clustering at scale. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

Yoshua Bengio and Jean-Sébastien Senécal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 2008.

Alina Beygelzimer, Sham Kakade, and John Langford. Cover trees for nearest neighbor. In *International conference on Machine learning (ICML)*, 2006.

Rajarshi Bhowmik, Karl Stratos, and Gerard de Melo. Fast and effective biomedical entity linking using a dual encoder. *arXiv preprint arXiv:2103.05028*, 2021.

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

Haiyan Cai. Exact bound for the convergence of metropolis chains. *Stochastic analysis and applications*, 2000.

Jason Chang and John W Fisher III. Parallel sampling of dp mixture models using sub-cluster splits. *Advances in Neural Information Processing Systems*, 26, 2013.

Anna Choromanska and Claire Monteleoni. Online clustering with experts. *Artificial Intelligence and Statistics (AISTATS)*, 2012.

Anna E Choromanska and John Langford. Logarithmic time online multiclass prediction. *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Ryan Curtin, William March, Parikshit Ram, David Anderson, Alexander Gray, and Charles Isbell. Tree-independent dual-tree algorithms. In *International Conference on Machine Learning (ICML)*, 2013.

Ryan R Curtin. A dual-tree algorithm for fast k-means clustering with large k. In *Proceedings of the 2017 SIAM International Conference on Data Mining (SDM)*. SIAM, 2017.

Ryan R Curtin, Dongryeol Lee, William B March, and Parikshit Ram. Plug-and-play dual-tree algorithm runtime analysis. *Journal of Machine Learning Research (JMLR)*, 2015.

Zhuyun Dai, Vincent Y Zhao, Ji Ma, Yi Luan, Jianmo Ni, Jing Lu, Anton Bakalov, Kelvin Guu, Keith B Hall, and Ming-Wei Chang. Promptagator: Few-shot dense retrieval from 8 examples. *arXiv preprint arXiv:2209.11755*, 2022.

Hal Daumé III, Nikos Karampatziakis, John Langford, and Paul Mineiro. Logarithmic time one-against-some. In *International Conference on Machine Learning (ICML)*, 2017.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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.

Laxman Dhulipala, David Eisenstat, Jakub Lacki, Vahab Mirrokni, and Jessica Shi. Hierarchical agglomerative graph clustering in poly-logarithmic depth. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Laxman Dhulipala, David Durfee, Janardhan Kulkarni, Richard Peng, Saurabh Sawlani, and Xiaorui Sun. Parallel batch-dynamic graphs: Algorithms and lower bounds. In *Symposium on Discrete Algorithms (SODA)*, 2020.

Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web*, pages 577–586, 2011.

Marc Dymetman, Guillaume Bouchard, and Simon Carter. The os* algorithm: a joint approach to exact optimization and sampling. *arXiv preprint arXiv:1207.0742*, 2012.

Yury Elkin and Vitaliy Kurlin. A new compressed cover tree guarantees a near linear parameterized complexity for all k-nearest neighbors search in metric spaces. *arXiv preprint arXiv:2111.15478*, 2021.

Nicholas FitzGerald, Dan Bikel, Jan Botha, Daniel Gillick, Tom Kwiatkowski, and Andrew McCallum. MOLEMAN: Mention-only linking of entities with a mention annotation network. In *Proceedings of the Association for Computational Linguistics and the International Joint Conference on Natural Language Processing (ACL-IJCNLP)*, 2021.

Luyu Gao and Jamie Callan. Condenser: a pre-training architecture for dense retrieval. *arXiv preprint arXiv:2104.08253*, 2021.

Dan Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. In *Conference on Computational Natural Language Learning (CoNLL)*, 2019.

Alex Gittens and Michael Mahoney. Revisiting the nystrom method for improved large-scale machine learning. In *International Conference on Machine Learning (ICML)*, 2013.

Yan Gu, Zachary Napier, Yihan Sun, and Letong Wang. Parallel cover trees and their applications. In *Proceedings of the ACM Symposium on Parallelism in Algorithms and Architectures*, 2022.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International Conference on Machine Learning (ICML)*, 2020.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017.

Jacob Holm, Kristian De Lichtenberg, and Mikkel Thorup. Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM (JACM)*, 2001.

Iris AM Huijben, Wouter Kool, Max Benedikt Paulus, and Ruud JG Van Sloun. A review of the gumbel-max trick and its extensions for discrete stochasticity in machine learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Gautier Izacard, Patrick Lewis, Maria Lomeli, Lucas Hosseini, Fabio Petroni, Timo Schick, Jane Dwivedi-Yu, Armand Joulin, Sebastian Riedel, and Edouard Grave. Few-shot learning with retrieval augmented language models. *arXiv preprint arXiv:2208.03299*, 2022.

Mike Izbicki and Christian Shelton. Faster cover trees. In *International Conference on Machine Learning (ICML)*, 2015.

Prateek Jain, Brian Kulis, Inderjit Dhillon, and Kristen Grauman. Online metric learning and fast similarity search. *Advances in neural information processing systems*, 21, 2008.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kalina Jasinska-Kobus, Marek Wydmuch, Devanathan Thiruvenkatachari, and Krzysztof Dembczynski. Online probabilistic label trees. In *Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2021.

Yacine Jernite, Anna Choromanska, and David Sontag. Simultaneous learning of trees and representations for extreme classification and density estimation. In *International Conference on Machine Learning (ICML)*, 2017.

Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of ACM SIGIR conference on research and development in Information Retrieval*, 2020.

Ari Kobren, Nicholas Monath, Akshay Krishnamurthy, and Andrew McCallum. A hierarchical algorithm for extreme clustering. *Conference on Knowledge Discovery and Data Mining (KDD)*, 2017.

Robert Krauthgamer and James R Lee. Navigating nets: Simple algorithms for proximity search. In *Symposium on Discrete algorithms (SODA)*, 2004.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. Sampling methods for the nyström method. *The Journal of Machine Learning Research*, 2012.

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. Natural Questions: A Benchmark for Question Answering Research. *Transactions of the Association for Computational Linguistics (TACL)*, 2019.

Edo Liberty, Ram Sriharsha, and Maxim Sviridenko. An algorithm for online k-means clustering. *Algorithm Engineering and Experiments (ALENEX)*, 2016.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

Lajanugen Logeswaran, Ming-Wei Chang, Kenton Lee, Kristina Toutanova, Jacob Devlin, and Honglak Lee. Zero-shot entity linking by reading entity descriptions. In *Association for Computational Linguistics (ACL)*, 2019.

Chris J Maddison, Andriy Mnih, and Yee Whye Teh. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Yury Malkov, Alexander Ponomarenko, Andrey Logvinov, and Vladimir Krylov. Approximate nearest neighbor al-

gorithm based on navigable small world graphs. *Information Systems*, 2014.

Kerrie L Mengersen and Richard L Tweedie. Rates of convergence of the hastings and metropolis algorithms. *The annals of Statistics*, 1996.

Aditya Krishna Menon, Anand Rajagopalan, Baris Sumengen, Gui Citovsky, Qin Cao, and Sanjiv Kumar. Online hierachical clustering approximations. *arXiv Pre-print*, 2019.

Benjamin Moseley, Kefu Lu, Silvio Lattanzi, and Thomas Lavastida. A framework for parallelizing hierarchical clustering methods. *European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD)*, 2019.

Radford M Neal. *Probabilistic inference using Markov chain Monte Carlo methods*. Department of Computer Science, University of Toronto Toronto, ON, Canada, 1993.

Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 2000.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*, 2021.

Max Paulus, Dami Choi, Daniel Tarlow, Andreas Krause, and Chris J Maddison. Gradient estimation with stochastic softmax tricks. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Yujie Qian, Jinhyuk Lee, Sai Meher Karthik Duddu, Zhuyun Dai, Siddhartha Brahma, Iftekhar Naim, Tao Lei, and Vincent Y Zhao. Multi-vector retrieval as sparse alignment. *arXiv preprint arXiv:2211.01267*, 2022.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 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.

Ankit Singh Rawat, Jiecao Chen, Felix Xinnan X Yu, Ananda Theertha Suresh, and Sanjiv Kumar. Sampled softmax with random fourier features. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

Ankit Singh Rawat, Aditya Krishna Menon, Andreas Veit, Felix Yu, Sashank J Reddi, and Sanjiv Kumar. Doubly-stochastic mining for heterogeneous retrieval. *arXiv preprint arXiv:2004.10915*, 2020.

Sashank J. Reddi, Satyen Kale, Felix Yu, Daniel Holtmann-Rice, Jiecao Chen, and Sanjiv Kumar. Stochastic negative mining for learning with large output spaces. In *Proceedings of the Twenty-Second International Conference on Artificial Intelligence and Statistics*, 2019.

Ruiyang Ren, Yingqi Qu, Jing Liu, Wayne Xin Zhao, Qiaoqiao She, Hua Wu, Haifeng Wang, and Ji-Rong Wen. Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking. *arXiv preprint arXiv:2110.07367*, 2021.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *arXiv preprint arXiv:2112.01488*, 2021.

Daniel Dominic Sleator and Robert Endre Tarjan. A data structure for dynamic trees. In *Symposium on the Theory of Computing (STOC)*, 1981.

Wen Sun, Alina Beygelzimer, Hal Daumé Iii, John Langford, and Paul Mineiro. Contextual memory trees. In *International Conference on Machine Learning (ICML)*, 2019.

Alexander Terenin, David R Burt, Artem Artemev, Seth Flaxman, Mark van der Wilk, Carl Edward Rasmussen, and Hong Ge. Numerically stable sparse gaussian processes via minimum separation using cover trees. *arXiv preprint arXiv:2210.07893*, 2022.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.

Thomas Tseng, Laxman Dhulipala, and Guy Blelloch. Batch-parallel euler tour trees. In *2019 Proceedings of the Twenty-First Workshop on Algorithm Engineering and Experiments (ALENEX)*, 2019.

George Tucker, Andriy Mnih, Chris J Maddison, John Lawson, and Jascha Sohl-Dickstein. Rebar: Low-variance, unbiased gradient estimates for discrete latent variable models. *Advances in Neural Information Processing Systems*, 30, 2017.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems (NeurIPS)*, 2017.

Shankar Vembu, Thomas Gärtner, and Mario Boley. Probabilistic structured predictors. In *Uncertainty in Artificial Intelligence (UAI)*, 2009.

Christopher Williams and Matthias Seeger. Using the nyström method to speed up kernel machines. *Advances in neural information processing systems (NeurIPS)*, 2000.

Le Wu, Yonghui Yang, Kun Zhang, Richang Hong, Yanjie Fu, and Meng Wang. Joint item recommendation and attribute inference: An adaptive graph convolutional

network approach. In *ACM SIGIR conference on research and development in Information Retrieval (SIGIR)*, 2020.

Ledell Wu, Fabio Petroni, Martin Josifoski, Sebastian Riedel, and Luke Zettlemoyer. Scalable zero-shot entity linking with dense entity retrieval. *arXiv preprint arXiv:1911.03814*, 2019.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations (ICLR)*, 2020.

Yuanhao Xiong, Wei-Cheng Chang, Cho-Jui Hsieh, Hsiang-Fu Yu, and Inderjit Dhillon. Extreme Zero-Shot learning for extreme text classification. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, 2022.

Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. Efficient passage retrieval with hashing for open-domain question answering. *arXiv preprint arXiv:2106.00882*, 2021.

Hsiang-Fu Yu, Kai Zhong, Jiong Zhang, Wei-Cheng Chang, and Inderjit S Dhillon. Pecos: Prediction for enormous and correlated output spaces. *Journal of Machine Learning Research*, 2022.

Manzil Zaheer, Michael Wick, Jean-Baptiste Tristan, Alex Smola, and Guy Steele. Exponential stochastic cellular automata for massively parallel inference. In *Artificial Intelligence and Statistics*, 2016.

Manzil Zaheer, Satwik Kottur, Amr Ahmed, José Moura, and Alex Smola. Canopy fast sampling with cover trees. In *International Conference on Machine Learning*, 2017.

Manzil Zaheer, Guru Guruganesh, Golan Levin, and Alex Smola. Terrapattern: A nearest neighbor search service. *Pre-print*, 2019.

Huayi Zhang, Lei Cao, Yizhou Yan, Samuel Madden, and Elke A. Rundensteiner. Continuously adaptive similarity search. In *Proceedings of the ACM SIGMOD International Conference on Management of Data (SIGMOD)*, 2020.

Supplemental Material: Improving Dual-Encoder Training through Dynamic Indexes for Negative Mining

A EFFICIENT AND ACCURATE SAMPLES FROM THE SOFTMAX DISTRIBUTION

A.1 Rejection Sampling

Rejection sampling for the softmax distribution follows closely the proof in Zaheer et al. (2017) for mixture models.

We sample from:

$$y \sim \frac{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}{\sum_{y'} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle)}$$

and accept with probability

$$e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}.$$

If we have

$$e^{-\epsilon_r} \leqslant \frac{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)} \leqslant e^{\epsilon_r}. \tag{12}$$

Then, if we want determine the probability of sampling a particular target, y, denoted $\Pr(y)$. Producing y can be done by sampling and accepting y or and sampling and rejecting another target y' and then accepting y in one of the subsequent rounds of sampling.

$$
\begin{aligned}
\Pr(y) &= \frac{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y})\rangle)} + \Pr(y) \sum_{y'\in\mathcal{Y}} \left(1 - e^{-\epsilon_r}\frac{\exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle)}\right) \frac{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} \\
&= e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} + \frac{\Pr(y)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} \sum_{y'\in\mathcal{Y}} \left(1 - e^{-\epsilon_r}\frac{\exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle)}\right) \exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle) \\
&= e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} + \frac{\Pr(y)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} \sum_{y'\in\mathcal{Y}} \left(\exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle) - e^{-\epsilon_r}\exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)\right) \\
&= e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} + \frac{\Pr(y)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} \left(\sum_{y'\in\mathcal{Y}} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y'})\rangle) - e^{-\epsilon_r} \sum_{y'\in\mathcal{Y}} \exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)\right) \\
&= e^{-\epsilon_r} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} + \Pr(y) - \frac{\Pr(y)}{\sum_{y''} \exp(\beta\langle f_\theta(x), f_\phi(\hat{y''})\rangle)} e^{-\epsilon_r} Z \\
\Pr(y) &= \frac{1}{Z} \exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)
\end{aligned}
\tag{13}
$$

And so the rejection sampling strategy will sample from the true softmax distribution.

A.2 Metropolis-Hastings Approximate Softmax

A common approach for providing samples from a distribution that is difficult to sample from is the Metropolis-Hastings algorithm. Recall that Metropolis-Hastings produces a sample from $P(y|x)$ by iteratively sampling a state change from a proposal distribution, denoted Q, and determines whether or not to 'accept' the state change based on an *acceptance ratio*. In particular, we will use an *independent* Metropolis-Hastings method. That is the proposal distribution Q is independent of the current state of the Markov chain. For the t-state in the chain we have a proposal distribution of the form $y^{(t)} \sim Q(y|x, y^{(t-1)}) = Q(y|x)$. The acceptance ratio is defined as:

$$A(y^{(t-1)}, y^{(t)}) = \min\left(1, \frac{P(y^{(t)}|x)}{P(y^{(t-1)}|x)} \frac{Q(y^{(t-1)}|x)}{Q(y^{(t)}|x)}\right) \tag{14}$$

A.3 Proofs

Proposition 1. *Under Assumption 3, given the clustering at level ℓ, $\mathcal{C}_{(\ell)}$, approximating y with the cluster representative, $y^{(\mathcal{C}_{(\ell)})}$, satisfies the following with $\epsilon_r = \beta \cdot b^\ell$:*

$$e^{-\epsilon_r} \leqslant \frac{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)} \leqslant e^{\epsilon_r}. \tag{8}$$

Proof.

$$\frac{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C})})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)} \leqslant \exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C})}) - f_\phi(y)\rangle) \tag{15}$$

$$\leqslant \exp(\beta||f_\theta(x)||_2 ||f_\phi(y^{(\mathcal{C})}) - f_\phi(y)||_2) \qquad \rhd \text{Cauchy-Schwartz} \tag{16}$$

$$\tag{17}$$

$$\leqslant \exp(\beta||f_\phi(y^{(\mathcal{C})}) - f_\phi(y)||_2) \qquad \rhd \text{Assumption 3} \tag{18}$$

$$\leqslant \exp(\beta b^\ell) \qquad \rhd \text{Covering Property.} \tag{19}$$

$\square$

Proposition 2. *Given Assumption 3, to achieve a maximum ratio of true softmax to proposal distribution equal to γ i.e., $\max_{y\in\mathcal{Y}} \frac{P(y|x)}{Q(y|x;\mathcal{C}_{(\ell)})} = \gamma$, we need the clustering at level ℓ, where: $\ell \triangleq \max\{\ell \in \mathbb{Z} : b^\ell \leqslant \frac{1}{2\beta}\log\gamma\}$.*

Proof. Let ℓ be the level of the clustering $\mathcal{C}$ used.

$$\max_{y\in\mathcal{Y}} \frac{P(y|x)}{Q(y|x;\mathcal{C}_{(\ell)})} = \max_{y\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle)} \cdot \frac{\hat{Z}}{Z} \tag{20}$$

$$\leqslant \max_{y\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle)} \cdot \max_{y'\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y'^{(\mathcal{C}_{(\ell)})})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)} \tag{21}$$

The above inequality follows by property of mediant. Now define:

$$R_1 \triangleq \max_{y\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle)} \tag{22}$$

$$R_2 \triangleq \max_{y'\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y'^{(\mathcal{C}_{(\ell)})})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y')\rangle)}. \tag{23}$$

Observe for R_1:

$$\max_{y\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle)} = \max_{y\in\mathcal{Y}} \exp(\beta\langle f_\theta(x), f_\phi(y)\rangle - \beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle) \tag{24}$$

$$= \max_{y\in\mathcal{Y}} \exp(\beta\langle f_\theta(x), f_\phi(y) - f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle) \tag{25}$$

$$\leqslant \max_{y\in\mathcal{Y}} \exp(\beta||f_\theta(x)||_2 ||f_\phi(y) - f_\phi(y^{(\mathcal{C}_{(\ell)})})||_2) \tag{26}$$

$$\leqslant \max_{y\in\mathcal{Y}} \exp(\beta||f_\phi(y) - f_\phi(y^{(\mathcal{C}_{(\ell)})})||_2) \leqslant \exp(\beta b^\ell) \tag{27}$$

Similarly for R_2:

$$\max_{y\in\mathcal{Y}} \frac{\exp(\beta\langle f_\theta(x), f_\phi(y^{(\mathcal{C}_{(\ell)})})\rangle)}{\exp(\beta\langle f_\theta(x), f_\phi(y)\rangle)} \leqslant \max_{y\in\mathcal{Y}} \exp(\beta||f_\phi(y^{(\mathcal{C}_{(\ell)})}) - f_\phi(y)||_2) \leqslant \exp(\beta b^\ell) \tag{28}$$

Combining above two:

$$\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q(y|x; \mathcal{C})} \leqslant R_1 \times R_2 \leqslant \exp(\beta b^\ell) \exp(\beta b^\ell) = \exp(2\beta b^\ell) \tag{29}$$

Thus, to maintain maximum ratio γ, we need to select ℓ such that:

$$\exp(2\beta b^\ell) \leqslant \gamma \tag{30}$$

$$b^\ell \leqslant \frac{1}{2\beta} \log \gamma \tag{31}$$

which is as stated in the proposition. $\qquad\square$

Remark 1. *Observe the relationship between γ and ℓ. Descending one more level of the tree to level $\ell - 1$, reduces the ratio from γ (selecting level ℓ) to $\gamma^{\frac{1}{b}}$.*

Proof. Let,

$$b^\ell \leqslant \frac{1}{2\beta} \log \gamma \tag{32}$$

$$b^{\ell-1} \leqslant \frac{1}{2\beta} \log \gamma', \tag{33}$$

then,

$$\frac{1}{b} b^\ell \leqslant \frac{1}{b} \frac{1}{2\beta} \log \gamma \tag{34}$$

$$b^{\ell-1} \leqslant \frac{1}{2\beta} \log \gamma^{\frac{1}{b}}, \tag{35}$$

and so $\gamma' \leqslant \gamma^{\frac{1}{b}}$. $\qquad\square$

Proposition 3. *Algorithm 1 produces samples from the softmax $P(y|x)$ in time $\mathcal{O}(|\mathcal{C}_\ell| + \alpha^4 e^{\beta b^{\ell+2}})$ for cover trees and $\mathcal{O}(|\mathcal{C}_\ell| + \alpha^3 e^{\beta b^{\ell+2}})$ for SG trees.*

Proof. We want to show two properties of the rejection sampling algorithm, its running time and its correctness (e.g., that it samples from the true posterior).

The proof follows very closely to the analogous algorithm for mixture models by Zaheer et al. (2017).

First, we want to show that the running time is $\mathcal{O}(|\mathcal{C}_{(\ell)}| + \mathsf{BF}e^{b^\ell})$ where BF is the branching factor for cover/SG trees ($\mathcal{O}(\alpha^4)$ and $\mathcal{O}(\alpha^3)$ respectively). We begin by considering the expected number of rejections from the first level ℓ. Based on our definition of Q, the number of samples is upper bounded by $e^{\beta b^{\ell+2}}$, therefore the number of rejections $e^{\beta b^{\ell+2}} - 1$. If we consider the next level down, there would be $e^{\beta b^{\ell+1}} - 1$ rejections, the level after that $e^{\beta b^\ell} - 1$, $e^{\beta b^{\ell-1}} - 1$, etc. We can use the branching factor BF to give a bound on how expensive the sampling step is at every level. We are interested therefore in:

$$\mathsf{BF} \sum_{k=1}^{\infty} \left(e^{\beta b^{\ell-k}} - 1 \right). \tag{36}$$

We have that $e^x - 1 \leqslant xe^a$ for $x \in [0, a]$ and $\sum_{k=1}^{\infty} b^{-k} = 1$ and so:

$$\mathsf{BF} \sum_{k=1}^{\infty} \left(e^{\beta b^{\ell-k}} - 1 \right) \leqslant \mathsf{BF} \cdot e^{\beta b^\ell} \sum_{k=1}^{\infty} b^{-k} = \mathsf{BF} \cdot e^{\beta b^\ell} \tag{37}$$

We then need to consider the cost of the initial sampling step, which is not BF, but rather depends on the number of clusters in the partition $|\mathcal{C}_{(\ell)}|$, leading to $\mathcal{O}(|\mathcal{C}_{(\ell)}| + \mathsf{BF} \cdot e^{\beta b^\ell})$. It is important to note here that while the running time of the algorithm depends only on dual branching factor of the trees, the depth of the SG tree differs from that of the cover

tree. The depth of the SG tree is $\mathcal{O}(\log(\frac{d_{\max}}{d_{\min}}))$ where $d_{\max}$ is the largest pairwise distance and $d_{\min}$ the minimum pairwise distance (this ratio also know as the aspect ratio), whereas the depth of the cover tree is $\mathcal{O}(\alpha^2 \log |\mathcal{Y}|)$.

Next, let's consider the correctness of the method. We want to determine the probability of sampling a particular target y, denoted $\mathsf{Pr}(y)$. Recall that to sample y, we need to follow the path in the tree from level ℓ to the level in which y first appears as a cluster representative and accept that cluster. We will refer to this level in which y first appears as k. There is a path of selected nodes $\mathcal{C}_\ell, \mathcal{C}_{\ell-1}, \dots, \mathcal{C}_k$. Let $\mathcal{A}(y)$ be the probability of accepting y. To reduce the complexity of notation, define:

$$\mathsf{w}_\mathcal{C} = \exp(\beta \langle f_\theta(x), f_\phi(\mathcal{C}) \rangle) \tag{38}$$

We can write $\mathcal{A}(y)$ as:

$$\mathcal{A}(y) = \frac{1}{Z_\ell} e^{\beta b^\ell} \blacktriangle_{\mathcal{C}_\ell} \mathsf{w}_{\mathcal{C}_\ell} \qquad\qquad \triangleright \text{ Sample at top level} \tag{39}$$

$$\times \left[\prod_{j=k+1}^{\ell} \left(1 - \frac{\frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_j}}{e^{\beta b^j} \blacktriangle_{\mathcal{C}_j} \mathsf{w}_{\mathcal{C}_j}} \right) \frac{e^{\beta b^{j-1}} \blacktriangle_{\mathcal{C}_{j-1}} \mathsf{w}_{\mathcal{C}_{j-1}}}{e^{\beta b^j} \blacktriangle_{\mathcal{C}_j} \mathsf{w}_{\mathcal{C}_j} - \frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_j}} \right] \quad \triangleright \text{ Rejecting and selecting path to } \mathcal{C}_k \tag{40}$$

$$\times \frac{\frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_k}}{e^{\beta b^k} \blacktriangle_{\mathcal{C}_k} \mathsf{w}_{\mathcal{C}_k}} \qquad\qquad \triangleright \text{ Accepting the given cluster } \mathcal{C}_k \tag{41}$$

$$= \frac{1}{Z_\ell} e^{\beta b^\ell} \blacktriangle_{\mathcal{C}_\ell} \mathsf{w}_{\mathcal{C}_\ell} \left[\prod_{j=k+1}^{\ell} \frac{e^{\beta b^{j-1}} \blacktriangle_{\mathcal{C}_{j-1}} \mathsf{w}_{\mathcal{C}_{j-1}}}{e^{\beta b^j} \blacktriangle_{\mathcal{C}_j} \mathsf{w}_{\mathcal{C}_j}} \right] \frac{\frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_k}}{e^{\beta b^k} \blacktriangle_{\mathcal{C}_k} \mathsf{w}_{\mathcal{C}_k}} \tag{42}$$

$$= \frac{1}{Z_\ell} e^{\beta b^\ell} \blacktriangle_{\mathcal{C}_\ell} \mathsf{w}_{\mathcal{C}_\ell} \left[\frac{e^{\beta b^j} \blacktriangle_{\mathcal{C}_j} \mathsf{w}_{\mathcal{C}_j}}{e^{\beta b^j} \blacktriangle_{\mathcal{C}_j} \mathsf{w}_{\mathcal{C}_j}} \right] \frac{\frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_k}}{e^{\beta b^k} \blacktriangle_{\mathcal{C}_k} \mathsf{w}_{\mathcal{C}_k}} \tag{43}$$

$$= \frac{1}{Z_\ell} \frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_k} \tag{44}$$

Now, we have to consider the probability $\mathcal{R}$ of restarting the sampler, e.g.,

$$\mathcal{R} = 1 - \sum_{y'} \mathcal{A}(y') = 1 - \sum_{y'} \frac{1}{Z_\ell} \frac{1}{|\mathcal{Y}|} \mathsf{w}_{\mathcal{C}_k} = 1 - \frac{1}{|\mathcal{Y}|} \frac{Z}{Z_\ell} \tag{45}$$

Finally, consider $\mathsf{Pr}(y)$:

$$\mathsf{Pr}(y) = \mathcal{A}(y) + \mathsf{Pr}(y) \cdot \mathcal{R} \tag{46}$$

$$= \frac{1}{Z_\ell} \frac{1}{|\mathcal{Y}|} \mathsf{w}_y + \mathsf{Pr}(y) - \mathsf{Pr}(y) \frac{1}{|\mathcal{Y}|} \frac{Z}{Z_\ell} \tag{47}$$

$$\mathsf{Pr}(y) = \frac{1}{Z} \mathsf{w}_y \tag{48}$$

Therefore, the algorithm samples from the true softmax. $\qquad\square$

Proposition 4. *Let $\mathcal{C}$ be the output of Algorithm 2, then $\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q(y|x, \mathcal{C})} \leqslant \gamma$ under Assumption 3.*

Proof. Recall that Proposition 2 ensures that the initial partition achieves $\max_{y \in \mathcal{Y}} \frac{P(y|x)}{Q(y|x; \mathcal{C}_{(\ell)})} = \gamma$. For any cluster in the partition discovered by the algorithm, $\mathcal{C} \in \mathcal{C}$, let $k < \ell$ be level of the cluster. We therefore have:

$$\max_{y \in \mathcal{C}} \frac{P(y|x)}{Q(y|x; \mathcal{C})} \leqslant \exp(\beta b^k) \leqslant \exp(\beta b^\ell). \tag{49}$$

The covering property and triangle inequality ensures that if $||f_\theta(x) - f_\phi(\mathcal{C})||_2 > b^k + b^m$, then every target in $\mathcal{C}$ at least b^m from $f_\theta(x)$.

$\qquad\square$

A.4 Gradient-Bias

Proposition 5. *Let P be the true softmax and Q_{MH} be the Metropolis-Hastings approximation to the softmax. Under Assumption 2, we have $||\mathbb{E}[\nabla_\Theta\hat{\mathcal{L}}] - \nabla_\Theta\mathcal{L}|| \leqslant 2\epsilon\beta M$, where $||P - Q_{\mathsf{MH}}||_{\mathsf{TV}} \leqslant \epsilon$.*

Proof. Observe that:

$$\nabla_\Theta\mathcal{L}(x_i, y_i) = -\beta\nabla_\Theta\langle f_\theta(x_i), f_\phi(y_i)\rangle + \beta\mathbb{E}_P[\nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle] \tag{50}$$

$$\mathbb{E}[\nabla_\Theta\hat{\mathcal{L}}(x_i, y_i)] = -\beta\nabla_\Theta\langle f_\theta(x_i), f_\phi(y_i)\rangle + \beta\mathbb{E}_{Q_{\mathsf{MH}}}[\nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle] \tag{51}$$

$$\mathbb{E}[\nabla_\Theta\hat{\mathcal{L}}(x_i, y_i)] - \nabla_\Theta\mathcal{L}(x_i, y_i) = \beta\mathbb{E}_{Q_{\mathsf{MH}}}[\nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle] - \beta\mathbb{E}_P[\nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle] \tag{52}$$

$$\mathbb{E}[\nabla_\Theta\hat{\mathcal{L}}(x_i, y_i)] - \nabla_\Theta\mathcal{L}(x_i, y_i) = \beta\sum_y \nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle(P(y|x) - Q_{\mathsf{MH}}(y|x)) \tag{53}$$

Now using the bound on the total variation:

$$||\mathbb{E}[\nabla_\Theta\hat{\mathcal{L}}(x_i, y_i)] - \nabla_\Theta\mathcal{L}(x_i, y_i)|| = \left\|\beta\sum_y \nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle(P(y|x) - Q_{\mathsf{MH}}(y|x))\right\| \tag{54}$$

$$= \beta\sum_y ||\nabla_\Theta\langle f_\theta(x_i), f_\phi(y)\rangle|| \cdot |(P(y|x) - Q_{\mathsf{MH}}(y|x)| \tag{55}$$

$$\leqslant \beta\sum_y M|(P(y|x) - Q_{\mathsf{MH}}(y|x)| \tag{56}$$

$$= \beta M\|P - Q_{\mathsf{MH}}\|_1 \tag{57}$$

$$= 2\beta M\|P - Q_{\mathsf{MH}}\|_{\mathsf{TV}} \tag{58}$$

$$\leqslant 2\beta M\epsilon \tag{59}$$

$\square$

A.5 Dynamic Maintenance of the Tree Structure

Proposition 6 *Under Assumptions 1,2,3, let ϕ_t and ϕ_{t+w} refer to the model parameters and model parameters after w more steps of gradient descent with learning rate η.*

$$\left|||f_{\phi_t}(y) - f_{\phi_t}(y')||_2 - ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||_2\right| \leqslant 4w\eta\beta LM. \tag{60}$$

Proof. We analyze this with similar techniques and assumptions as Lindgren et al. (2021). First consider a bound on the gradient norm. Let k by the number of negative samples where the negative samples are given by $N_s = \{y_{s_j} : j \in [k]\}$

$$||\nabla_\Theta\hat{\mathcal{L}}(x_i, y_i)|| = \left\|-\beta\nabla_\Theta\langle f_\theta(x_i), f_\phi(y_i)\rangle + \beta\frac{1}{k}\sum_{j=1}^k \nabla_\Theta\langle f_\theta(x_i), f_\phi(y_{y_{s_j}})\rangle\right\| \tag{61}$$

$$\leqslant \beta||\nabla_\Theta\langle f_\theta(x_i), f_\phi(y_i)\rangle|| + \beta\frac{1}{k}\left\|\sum_{j=1}^k \nabla_\Theta\langle f_\theta(x_i), f_\phi(y_{y_{s_j}})\rangle\right\| \tag{62}$$

$$\leqslant \beta||\nabla_\Theta\langle f_\theta(x_i), f_\phi(y_i)\rangle|| + \beta\frac{1}{k}\sum_{j=1}^k \left\|\nabla_\Theta\langle f_\theta(x_i), f_\phi(y_{y_{s_j}})\rangle\right\| \tag{63}$$

$$\leqslant 2\beta M \tag{64}$$

Now, consider the difference of the dual encoder parameters at timestep, t, Θ_t and at timestep, $t + w$, Θ_{t+w}.

$$\Theta_t - \Theta_{t+w} = \sum_{i=t}^{t+w} \eta \nabla_\Theta \hat{\mathcal{L}}(x_i, y_i) \tag{65}$$

$$||\Theta_t - \Theta_{t+w}|| = \left\|\sum_{i=t}^{t+w} \eta \nabla_\Theta \hat{\mathcal{L}}(x_i, y_i)\right\| \tag{66}$$

$$\leqslant \sum_{i=t}^{t+w} \eta \left\|\nabla_\Theta \hat{\mathcal{L}}(x_i, y_i)\right\| \tag{67}$$

$$= w\eta 2\beta M \tag{68}$$

Applying the triangle inequality

$$\left|||f_{\phi_t}(y) - f_{\phi_t}(y')|| - ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||\right| \tag{69}$$

$$\leqslant \left|||f_{\phi_t}(y) - f_{\phi_{t+w}}(y)|| + ||f_{\phi_{t+w}}(y) - f_{\phi_t}(y')|| - ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||\right| \tag{70}$$

$$\leqslant \left|||f_{\phi_t}(y) - f_{\phi_{t+w}}(y)|| + ||f_{\phi_{t+w}}(y') - f_{\phi_t}(y')|| + ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')|| - ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||\right| \tag{71}$$

$$= \left|||f_{\phi_t}(y) - f_{\phi_{t+w}}(y)|| + ||f_{\phi_t}(y') - f_{\phi_{t+w}}(y')||\right| \tag{72}$$

Recall that the dual encoders satifies the Lipschitz assumption (Assumption 1):

$$||f_{\phi_t}(y) - f_{\phi_{t+w}}(y)|| \leqslant L||\Theta - \Theta'|| \leqslant w\eta 2\beta LM \quad \forall y \tag{73}$$

And so:

$$\left|||f_{\phi_t}(y) - f_{\phi_t}(y')|| + ||f_{\phi_{t+w}}(y) - f_{\phi_{t+w}}(y')||\right| \leqslant 4w\eta\beta LM. \tag{74}$$

$\square$

B   EFFICIENT RE-ENCODING OF TARGETS

In this section, we describe in more detail the re-encoding part of DyNNIBAL. Let's recap the structure of the re-encoding model. Recall that we keep a cached, low-dimensional version of each encoded target. We denote this cache as $Y'_t \subset \mathbb{R}^{d'}$ where d' is the number of landmarks used in Nyström / the reduced dimensionality and t represents the number of gradient steps. Note that while we denote separate caches for each time step, we need not physically store multiple such caches. The cache from the previous time step can be overwritten during the update.

We refer to Y_t as the set of target embeddings in their full dimensional space. This set Y_t is never fully instantiated. We have a set of d' *landmark* points, which are kept fixed during training and which were sampled uniformly at random. We will store these landmarks in their full dimensional form. We denote the set of landmarks as $\mathcal{S}$. We also have a set of training points used to fit the regression model, $\mathcal{R}$, which maps target embeddings in Y_t to form Y_{t+w}. Each time we fit $\mathcal{R}$, we sample s' targets uniformly at random and build a training dataset for the regressor using the updated dual encoder, $f_{\phi_{t+w}}$, specifically, $(\vec{y_1}, f_{\phi_{t+w}}(y_1)), \ldots (\vec{y_{s'}}, f_{\phi_{t+w}}(y_{s'}))$. We only need to store the regression training dataset points and the landmarks in their full dimension form. For all other targets, we can compute their approximate full dimensional representation using $\mathcal{R}$ and then immediately project into lower dimensional space d' using the landmarks/Nyström. We define the regression model $\mathcal{R}$ to be a low-rank, Nyström, regression model with the same landmarks $\mathcal{S}$. Therefore, the input to the regression model can be the low-dimensional representations.

In summary, the approximate re-encoding procedure would, re-encode s' points using the new encoder model, $f_{\phi_{t+w}}$, build a training dataset for the regression model $\mathcal{R}$, fit the regression model $\mathcal{R}$, use $\mathcal{R}$ to update the landmark point representation $\mathcal{S}$, use $\mathcal{R}$ and $\mathcal{S}$ to build the low-dimensional embeddings for all targets in Y'_t as well as the corresponding $(\mathbf{S}^T \mathbf{K} \mathbf{S})^{-1}$ projection matrix. This is summarized in Algorithm 5 (with initialization in Algorithm 6), which is called as a subroutine of the overall training algorithm (Algorithm 4).

It is important to note that we do not use these approximate re-encoded targets at evaluation time. We only use these dimensionality reduced targets at training time.

Lastly, we note that while we sample the landmark points from only the targets empirically, one could (in a more principled way) sample from all datapoints and targets.

Algorithm 5 Approximate Re-Encode

1: Let $\mathcal{S}$ refer to the set of landmark points
2: Sample s' targets uniformly at random from the collection of targets to form S'_t.
3: Let $Y'_{S'_t}$ be the low dimensional representations of the training points.
4: Build regression training dataset $(\vec{y_1}, f_{\phi_{t+w}}(y_1)), \ldots (\vec{y_{s'}}, f_{\phi_{t+w}}(y_{s'}))$ using the updated encoder model.
5: Train $\mathcal{R}$ on $(\vec{y_1}, f_{\phi_{t+w}}(y_1)), \ldots (\vec{y_{s'}}, f_{\phi_{t+w}}(y_{s'})) \triangleright \mathcal{R}$ is a low-rank regression model using the same landmarks $\mathcal{S}$.
6: Update the embedding of landmark representatives $\mathcal{S}$ using $\mathcal{R}$. Update Nyström projection matrix $(\mathbf{S}^T\mathbf{KS})^{-1}$
7: Produce new low-dimensional embeddings for all targets Y'_{t+w} using $\mathcal{R}$.

Algorithm 6 Initial Target Encoding

1: Sample landmarks $\mathcal{S}$ uniformly at random
2: Encode landmarks, fit Nyström.
3: Encode all of the targets with the encoder model f_ϕ and use Nyström to produce low dimensional representations Y'_0

B.1 Building SG Trees with Nyström Representations

When we use our Nyström-based lower dimensional representations, the unit norm assumptions about the targets no longer apply. At query time, for a given point x, we can think about it as corresponding to some row i in the $\mathbf{KS}$ matrix. Similarly each target y to some column j in $(\mathbf{S}^T\mathbf{KS})^{-1}\mathbf{S}^T\mathbf{K}$. This corresponds to using the $(\mathbf{S}^T\mathbf{KS})^{-1}\mathbf{S}^T\mathbf{K}$ representations as the target cluster representatives in the tree structure. However, when building (or re-building) the tree structure, we need to be able to measure the similarity between two targets, i.e., using both $\mathbf{KS}$ and $(\mathbf{S}^T\mathbf{KS})^{-1}\mathbf{S}^T\mathbf{K}$. We only need to use the $\mathbf{KS}$ representations during construction / re-building time however and if memory is a concern, we could only store a single representation and use the $(\mathbf{S}^T\mathbf{KS})^{-1}$ when measuring (dis)similarity.

Finally, we need to be able to use the SG Tree despite using inner product similarities (rather than a proper distance measure). Rather than the approach presented by Zaheer et al. (2019), we find that converting to distance by $\exp(-\mathbf{KS}(\mathbf{S}^T\mathbf{KS})^{-1}\mathbf{S}^T\mathbf{K})$ is more effective. We find that setting the base of the SG tree to be 1.05 seems to work well with this conversion of similarities to distances. Note that these modifications mean that even running the exact nearest neighbor search algorithm may result in approximations. However, we find empirically that this structure is sufficient for training dual encoders.

C ADDITIONAL EMPIRICAL DETAILS

The dual encoder models used in all experiments are transformers, initialized with the Roberta base model (Liu et al., 2019). We use the hyperparameters presented in Table 3. We note that the changes unique for Zeshel were done to account for the longer data point input length.

All models unit-norm the output embedding that is produced by the dual encoder. All models perform score-scaling as in Lindgren et al. (2021) with a scaling factor of 20.0. All models share sampled negative examples across all positives in the same batch. In-batch negatives are used by all models. Stochastic Negative mining and DyNNIBAL use uniform negatives in addition to the sampled hard negatives.

We first train models using uniform negatives and then apply DyNNIBAL. Note that this mimics the hypothesis that the softmax distribution would be closer to uniform in the beginning of training. We perform 4000 steps of uniform negatives for NQ and 8000 MSMARCO.

We use 128 landmarks in Nyström/regression model for all datasets. For Natural Questions and MSMARCO we use 8192 training examples for the regression model. For Zeshel we use 256 training examples for the regression model. For NQ, we perform a full refresh of all embeddings (re-initializing Nyström/regression model) once during training at step 7500.

We find that changing the number of landmarks used does not dramatically change the performance of the method. For instance, on Natural Questions (NQ), we find that in terms of R@1, 256 landmarks achieves 0.488 while with 128 landmarks gets 0.481 and 64 landmarks gets 0.479 and R@100 with 256 landmarks is 0.862 and 128 landmarks gets 0.859 and 64 landmarks gets 0.859.

Name	Value
Train Batch Size	128 total
Uniform Negatives	64 per training example (32 for Zeshel)
Sampled Negatives	64 per training example
Initial Warmup Learning Rate	1e-5
End Warmup Learning Rate	1e-7
Warmup Steps	10,000
Warmup Decay Type	Linear
Optimizer	Adam
Optimizer Beta1	0.9
Optimizer Beta2	0.999
Optimizer Epsilon	1e-8
Score Scaling	20.0
Max Data Point Feature Length	128 (256 for Zeshel)
Max Target Feature Length	256
Encoder Embedding Dimension	768

Table 3: Hyperparameters used for training dual encoder models.

Recall	Mem. %	R@1	R@10	R@100
In-batch Negatives	0%	0.388	0.726	0.861
Uniform Negatives	0%	0.393	0.713	0.851
Stochastic Neg	6.59%	0.413	0.737	0.865
DyNNIBAL	0.064%	0.421	0.742	0.870
Exhaustive	100%	0.444	0.756	0.880

Table 4: **Zeshel (Dev) Results**. We measure the recall performance on the entity linking dataset. We note that this dataset is considerably smaller than the other datasets in terms of number of targets. Still, we find that DyNNIBAL can achieve better performance than baseline methods while approaching the performance of the brute force oracle. DyNNIBAL achieves better performance that a Stochastic Negative Sampling baseline which uses more memory.

C.1 SG Tree Details

We use the Nyström-modified SG Tree that is described in §B.1. We notice that the tree construction and sampling procedure can be done much more efficiently on smaller trees. And so, rather than constructing one tree over all the targets, we construct a forest of 100 trees of roughly equal size. We construct and build samples from this forest independently and aggregate samples (by taking the max unnormalized probability).

D ZESHEL EXPERIMENTAL RESULTS

Zeshel (Logeswaran et al., 2019) is a dataset for classifying (linking) ambiguous mentions of entities to their unambiguous entity pages in Fandom Wikias. The original dataset separates individual Fandom Wikia into separate domains, which severely limits the number of targets. To make the task more challenging and better suited for evaluation of models approximating the softmax loss, we combine all Wikias together, resulting in a collection of 492K targets. We evaluate in terms of recall of the correct entity.

Tables 4 and 5 report the dev and test scores for the Zeshel dataset. Recall that the number of targets in this dataset is significantly less than the other two datasets (∼500K compared to 21M (Natural Questions), 8.8M (MSMARCO)). We find that the performance of DyNNIBAL improves upon baselines of Stochastic Negative Sampling and Uniform Negatives.

Because of the reduced size of the dataset, we use only 256 training points for the regression model. We also use a relatively small number of examples for stochastic negative mining (32768 in total), though this accounts for a larger memory percentage overall because of the reduced number of targets.

Recall	Mem. %	R@1	R@10	R@100
In-batch Negatives	0%	0.344	0.627	0.778
Uniform Negatives	0%	0.323	0.593	0.747
Stochastic Neg	6.59%	0.330	0.609	0.756
DyNNIBAL	0.064%	0.348	0.623	0.766
Exhaustive	100%	0.348	0.617	0.767

Table 5: **Zeshel (Test) Results**. We measure the recall performance on the entity linking dataset. While the dataset has considerably fewer targets, DyNNIBAL achieves nearly the same result as the exhaustive brute force method and better results than Stochastic Negative Mining and Uniform Negatives. In this setting, In-Batch Negatives performs very well, perhaps because of the relatively small number of targets.

E LIMITATIONS

The focus of this work is on improving the approximation of the cross entropy loss and the quality of the samples from the softmax distribution. For some tasks, it may be the case that different objectives, including multi-task, pretraining, contrastive, and others may lead to models that generalize better. After applying Nyström, approximate re-encoding step, and the sampling approximations, we lose theoretical properties. Future work could investigate both why these approximations work well and how to bound the approximation quality.

F ETHICAL CONSIDERATIONS

The proposed approach is subject to the same biases and ethical considerations as the dual encoders used as the base model. The reduction in error cannot be assumed to reduce or exacerbate the potentially negative aspects of such an encoder model. The biases and considerations of any such retrieval/classification task need to be carefully considered as with any model. The implications of negative sampling via uniform vs. hard methods will likely impact the decision boundary of the model (as observed in our empirical experiments). Understanding how the training method relates to biases in the data, labels, targets, would be an important consideration.

Symbol	Definition
x	A data point / the features of a data point, e.g. input to the data dual encoder.
$\mathcal{X}$	Space of data points
y	A target / the features of a target, e.g. input to the dual encoder.
$\mathcal{Y}$	Set of all targets
$\mathcal{L}$	Loss function, in particular cross entropy
$\hat{\mathcal{L}}$	Our approximate, sampling based loss function
L	Lipschitz value, Assumption 1
M	Bound on gradient of logits, Assumption 2
f_θ	The data point dual encoder
f_ϕ	The target dual encoder
Z	The partition function for the softmax. Eq 1.
$\hat{Z}$	The approximated partition function using the clustering-based approach. Eq 7.
Z_k	The normalizer constant when descending to level k in the rejection sampling approach.
Θ	The parameters of both the data and target dual encoders.
$P, P(y|x)$	True, exact softmax distribution
Q	The proposal distribution for Metropolis-Hastings. Eq 7.
Q_{MH}	The distribution over labels given by Metropolis Hastings sampling procedure. See Appendix A.2.
$e^{-\epsilon_{\mathsf{r}}}$	Used for unnormalized probability error term in rejection sampling
d	Dimensionality of dual encoder output.
$\mathcal{C}$	A clustering of the targets.
$y^{(\mathcal{C})}$	The cluster assignment of target y. Overloaded to also be the cluster's representative (or its features).
$\mathcal{C}$	A cluster of targets, e.g. $\mathcal{C} \subseteq \mathcal{Y}$.
$f_\phi(y^{(\mathcal{C})}), f_\phi(\mathcal{C})$	The encoded representation of the cluster's representative.
$\mathcal{T}$	A hierarchical clustering / cover or SG Tree of the targets
b	The base parameter of the cover / SG tree
$\mathsf{ch}(\mathcal{C})$	The children of a node in the hierarchical clustering/cover tree.
$\mathcal{C}_{(\ell)}$	The partition associated with level ℓ of the cover tree.
$Y_{(\ell)}$	The set of cluster representatives for the partition associated with level ℓ of the cover tree.
α	The expansion constant used in our theoretical analysis 4
$\mathsf{MAXD}(\mathcal{C})$	The maximum distance between the cluster representative of $\mathcal{C}$ and one of its descendants in an SG Tree.
$\mathsf{MIND}(\mathcal{C})$	The minimum distance between one of the pairs of children of $\mathcal{C}$ in an SG Tree.
γ	The upper bound on the ratio of the true softmax distribution, $\max_y \frac{P(y\|x)}{Q(y\|x;\mathcal{C})} \leqslant \gamma$
ρ	Used as γ^ρ to trade off between importance of R_1 and R_2.
w	Number of steps of gradient descent.
s	The Metropolis Hastings chain length
η	Gradient descent learning rate
d_{max}	Maximum pairwise distance among all pairs in a cover tree
d_{min}	Minimum pairwise distance among all pairs in a cover tree
$\mathbf{S}$	Binary matrix representing landmarks for Nyström
$\mathbb{S}$	Set of landmarks for Nyström
$\mathbf{K}$	Pairwise similarity matrix for Nyström
$\mathcal{R}$	Low-rank (Nyström) regression model to approximately re-encode the targets
$Y'_{S'_t}$	The low dimensional representations of the training points for $\mathcal{R}$. Not to be confused with $Y_{(\ell)}$, the representatives in a level of the cover tree.

Table 6: **Summary of Notation Used**