CONTEXTUAL DOCUMENT EMBEDDINGS

Anonymous authors
Paper under double-blind review

ABSTRACT

Dense document embeddings are central to neural retrieval. The dominant paradigm is to train and construct embeddings by running encoders directly on individual documents. In this work, we argue that these embeddings, while effective, are implicitly out-of-context for targeted use cases of retrieval, and that a contextualized document embedding should take into account both the document and neighboring documents in context - analogous to contextualized word embeddings. We propose two complementary methods for contextualized document embeddings: first, an alternative contrastive learning objective that explicitly incorporates the document neighbors into the intra-batch contextual loss; second, a new contextual architecture that explicitly encodes neighbor document information into the encoded representation. Results show that both methods achieve better performance than biencoders in several settings, with differences especially pronounced out-of-domain. We achieve state-of-the-art results on the MTEB benchmark with no hard negative mining, score distillation, dataset-specific instructions, intra-GPU example-sharing, or extremely large batch sizes. Our method can be applied to improve performance on any contrastive learning dataset and any biencoder.[1]

1 INTRODUCTION

Machine learning approaches to text retrieval aim to learn an embedded representation for indexing documents. Classically, this area was dominated by statistical approaches using sparse lexical matching methods based on n-gram frequencies such as BM25 (Robertson & Zaragoza, 2009). Only recently have neural networks become competitive with state-of-the-art models on retrieval tasks (Karpukhin et al., 2020; Thakur et al., 2021). The primary neural method is a *dual encoder* architecture that independently encodes both a document and query to a dense latent space for retrieval lookup. This document embedding space can improve upon a statistical model since it is learned end-to-end for retrieval.

However, there is at least one notable benefit of statistical approaches that is lost by neural models. Statistical models can easily incorporate prior corpus statistics such as inverse document frequency (IDF), into their representation. This prior term imparts context-dependence onto the model, since it can be updated based on information specific to retrieval in a given domain at test time. We contrast this contextual formulation with neural document encoders that are by definition a function of the document itself. For example consider the following document:

> The National Football League Draft is an annual event in which the National Football League (NFL) teams select eligible college football players...

Depending on the retrieval domain, e.g. Wikipedia search, sports articles, or televised events, IDF may weight terms such as `NFL`, `draft` or `annual` higher; a neural document embedding model would need to select a global weighting for this document.

In this work, we explore contextualization of document embeddings produced by dense encoders. The goal is to produce embeddings that are better able to handle retrieval tasks in specific challenging contexts. We propose two complementary changes to document encoders: a contextual training procedure and architecture.

[1]We plan to release our code and data for clustering, training, and inference as well as cluster indices for the corresponding datasets.

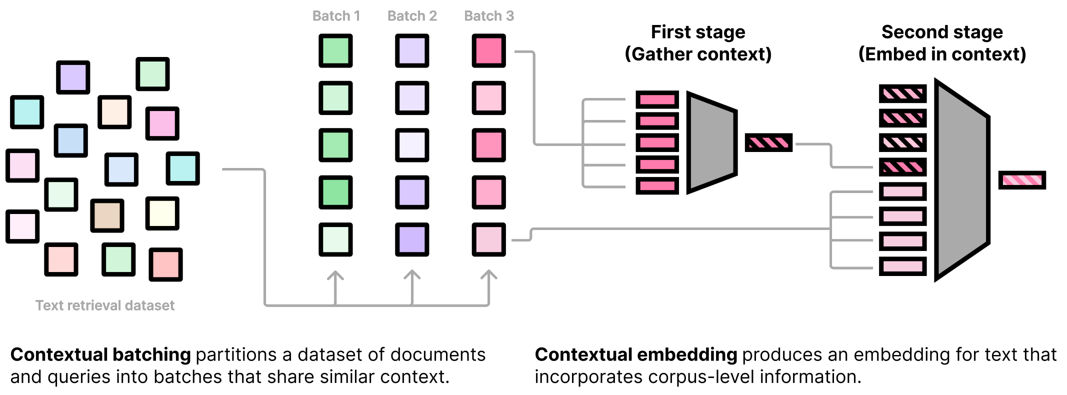
Batch 1
Batch 2
Batch 3
First stage (Gather context)
Second stage (Embed in context)
Text retrieval dataset
Contextual batching partitions a dataset of documents and queries into batches that share similar context.
Contextual embedding produces an embedding for text that incorporates corpus-level information.

Figure 1: Overview of our system for contextual document embeddings (CDE). Our model operates in two stages: a first stage used to characterize the dataset from samples, and a second stage used to embed the final document.

For contextual training, we aim to build a notion of neighboring documents directly into the contrastive learning process. We propose a method that uses on fast query-document clustering to produce a group of neighbors for each training batch. Each update for training is constructed purely from neighboring documents to ensure that embeddings can distinguish documents even in the most challenging contexts.

For the architecture, we propose a new encoder that injects information about the contextual documents during embedding. The proposed architecture augments the standard BERT-style encoder with additional conditioning that provides aggregated document-level information about neighboring documents. We call our method $\underline{\text{C}}$ontextual $\underline{\text{D}}$ocument $\underline{\text{E}}$mbedding (CDE). Analogously to pre-computed corpus-level statistics, this method provides a manner for the embedding to take into account the relative frequency of terms in context. The final output is still an embedding of the same size, so this does not require any additional storage or other changes to the retrieval process. When indexing, we utilize information from the corpus to produce document and query embeddings that are specific to a particular domain.

Experiments compare these two extensions to standard approaches for training document embeddings. Our results show that contextual contrastive training improves standard text embedding model training, and can be run without other approaches such as additional hard negatives. With the contextual encoder architecture, we see additional improvements over a baseline model in all settings tested, with larger improvements in highly specific domains such as small datasets of financial and medical documents. When trained at industry-scale, our model achieves state-of-the-art results for small (<250M parameter) models on the MTEB benchmark.

2 RELATED WORK

Text retrieval. Our work is related to the general field of text retrieval; we propose specific improvements to the training of "biencoder" text embedding models such as DPR (Karpukhin et al., 2020), GTR (Ni et al., 2021), Contriever (Izacard et al., 2022), LaPraDoR (Xu et al., 2022), Instructor (Su et al., 2023), Nomic-Embed (Nussbaum et al., 2024), E5 (Wang et al., 2024), and GTE (Li et al., 2023). We focus on the problem of adapting these text retrieval models to new corpora at test time; some prior work has noted this problem (Dai et al., 2022; Sciavolino, 2021) and proposed solutions such as unsupervised span-sampling and training on test corpora (Gao & Callan, 2021) and distillation on the test corpus from a reranker (Sung et al., 2023). Late interaction methods (Khattab & Zaharia, 2020; Santhanam et al., 2022) also offer one way to improve out-of-domain retrieval performance, but increase the runtime and complexity of search. We propose a better sampling scheme that can be used to train any biencoder or late interaction model as well as a *training-free* method for test-time adaptation.

Contrastive learning. Much research has focused on the effect of hard negatives on the performance of contrastive learning methods (Chen et al., 2020; Qu et al., 2021; Robinson et al., 2021; Wang et al., 2023). (Zhang & Stratos, 2021) observe that harder negatives provide a better approximation of the overall cross-entropy loss, but do not consider *batch*-level optimizations for negative selection. (Sachidananda et al., 2023) also consider contrastive batch sampling as a global optimization problem, but do not apply their technique to state-of-the-art transformer-based text embedding models. (Ma et al., 2024) use a clustering algorithm to partition a dataset into several sub-datasets, but train a different model on each sub-dataset. Our training algorithm aims to find the hardest possible batches to train text embedding model.

Test-time adaptation. Our method can be compared to other solutions to test-time adaptation, a problem that has been well-studied across a variety of domains (Jang et al., 2023). In retrieval, one form of test-time adaptation is pseudo-relevance feedback (PRF) (Rocchio, 1971; Li et al., 2018; Wang et al., 2021), where documents relevant to the query are used to construct a final, enhanced query representation. The query side of our model can be seen as a form of pseudo-relevance feedback; however, we train from scratch to support a more general form of PRF natively, on the document representation as well as the query.

Non-parametric modeling. Our contextual document model can be seen as a form of non-parametric modeling. This shows connections with the a large body of deep learning research such as the non-parametric transformer (NPT) (Kossen et al., 2022) and the subfield of Neural Processes (Garnelo et al., 2018; Kim et al., 2019; Nguyen & Grover, 2023). Semi-parametric models have been recently applied in NLP, specifically to the task of language modeling (Borgeaud et al., 2022; Khandelwal et al., 2020). Instead of using a retrieval model to build a semi-parametric langauge model, we build a semi-parametric model specifically for the task of retrieval.

3 Background

We can view text retrieval methods probabilistically as computing a distribution over potential documents based on a scalar score function $f(d, q)$ matching documents and queries:

$$p(d \mid q) = \frac{\exp f(d, q)}{\sum_{d' \in \mathcal{D}} \exp f(d', q)} \tag{1}$$

where $\mathcal{D}$ is a finite set of documents in a dataset. There is a wide variety of different definitions for f including full pairwise neural parameterizations (Nogueira & Cho, 2020). In this work, we focus on efficient retrieval methods using vector-based methods, also known as embedding models.

Vector retrieval methods assume that $f(d, q)$ can be factored into two embedding terms, $\phi(d) \cdot \psi(q)$, the document and query embedding respectively. This factorization allows precomputation of the document embeddings $\phi(d)$ for all $d \in \mathcal{D}$. This is critical for facilitating fast computation of $\arg\max_d p(d \mid q)$ or top-k variants (Douze et al., 2024).

In statistical retrieval, ϕ and ψ are closed-form functions of the data, often representing unigram or bigram counts by the relative frequency of word types. Notably for this work, these methods can also utilize distributional properties of the test dataset as a prior, for example through inverse document frequency (IDF). We represent this integration of dataset-level information by writing the vector product $\phi(d; \mathcal{D}) \cdot \psi(q; \mathcal{D})$.

In neural retrieval, we instead learn the representation as a dense vector. We assume access to a training corpus of document and query pairs (these may be supervised, i.e. gold-standard annotations, or unsupervised, i.e. noised synthetic examples), $\mathcal{D}_T = \{(d^1, q^1), ..., (d^J, q^J)\}$, with the aim of learning the embedding function ϕ and ψ.

Training can be motivated as maximizing likelihood of the document corresponding to each query, i.e. $\sum_j \log p(d^j \mid q^j)$. Unfortunately, since retrieval datasets can have $|\mathcal{D}|$ exceed millions of documents, computing the normalizer in Eq 1 at each training step is not an option. Instead contrastive learning is used where the likelihood is replaced with a biased approximation calculated from negative samples:

$$\max_{\phi,\psi} \sum_j \log p(d^j \mid q^j) \approx \sum_j \log \frac{\exp f(d^j, q^j)}{\sum_{d' \in \mathcal{H}(q^j)} \exp f(d', q^j)}$$

where $\mathcal{H}$ is a set of examples used to approximate the normalizing constant. In implementation, in addition to these hard negative examples, other examples from the mini-batch are also used to compute the normalizer since it requires no additional compute for calculating $\phi(d)$.

4 METHODS

In our work, we are interested in integrating contextual information into our embedding functions ϕ and ψ. The standard neural ϕ is purely a function of the document $\phi(d)$ and does not take into account any notion of context. This contrasts with the statistical model $\phi(\cdot; \mathcal{D})$ and $\psi(\cdot; \mathcal{D})$. Arguably this is not an issue if retrieval is completely in domain, as ϕ is capable of learning statistics such as IDF and average document length on the training set through gradient descent.

However, in many retrieval benchmarks, models are trained over a single set of documents $\mathcal{D}$ and then tested in many other domains $\mathcal{D}$ that differs significantly from $\mathcal{D}_T$. In this setting, training on $\mathcal{D}_T$ alone may not be able to provide robust embeddings when used in contexts such as $\mathcal{D}$.

4.1 CONTEXTUAL TRAINING WITH ADVERSARIAL CONTRASTIVE LEARNING

Returning to the example from the introduction, we assume that in a general purpose training corpus $\mathcal{D}_T$, the term `NFL` is a rare word appearing in relatively few documents and a useful signal. However, if at test time $\mathcal{D}$ is a corpus of sports articles, this word would be exceedingly common. Evaluation in this domain is, in a statistical sense, adversarial to the original dataset. To handle this issue, meta-learning-style objectives have shown to be effective for training document embedders. In these approaches, instead of sampling documents-query pairs iid, the objective first sample a domain and then sample a batch of examples. This ensures that the model mostly sees related training points in each domain.

We propose a training objective that synthesizes a large set of fine-grained domains to train the model on. Formally, our aim is to partition the training dataset $\mathcal{D}_T$ into groups $(\mathcal{B}^1, \dots \mathcal{B}^B)$ such that each group represents a self-similar pseudo-domain:

$$\max_{\phi,\psi} \sum_b \sum_{(d,q)\in\mathcal{B}^b} \log p(d \mid q) = \max_{\phi,\psi} \sum_b \sum_{(d,q)\in\mathcal{B}^b} \log \frac{\exp f(d,q)}{\sum_{(d',\cdot)\in\mathcal{B}^b} \exp f(d', q)}$$

Computationally, the inner term can be implemented as a single batch and computed efficiently without the need for separate hard negatives ($\mathcal{H}$). Ideally we want groups that are as challenging as possible. Zhang & Stratos (2021) show that increasing the partition term improves the contrastive approximation to the maximum likelihood the gradient. We can formalize the search for the most difficult configuration of batches as an optimization problem:

$$\max_{(\mathcal{B}^1,\dots\mathcal{B}^B)} \sum_b \sum_{\substack{(d,q)\in\mathcal{B}^b \\ (d',q')\in\mathcal{B}^b}} f(d,q') + f(d',q) = \max_{(\mathcal{B}^1,\dots\mathcal{B}^B)} \sum_b \sum_{\substack{(d,q)\in\mathcal{B}^b \\ (d',q')\in\mathcal{B}^b}} \phi(d)\cdot\psi(q') + \phi(d')\cdot\psi(q) \quad (2)$$

Solving this combinatorial objective exactly is intractable, but we can treat approximate a solution using clustering. We first move from a maximization to a minimization by replacing the two dot products with L_2, i.e. $m((d,q),(d',q')) = ||\phi(d) - \psi(q')|| + ||\phi('d) - \psi(q)||$ which is equivalent for normalized embeddings. We then note that treated as symmetric pairs, this term obeys the triangle inequality for any other pair m i.e:

$$m((d,q),m) + m(m,(d',q')) \geq m((d,q),(d',q'))$$

This implies that the following centroid-based objective represents an upper-bound on our original objective:

$$\min_{\substack{(\mathcal{B}^1,\ldots\mathcal{B}^B)\\(m^1,\ldots,m^B)}} \sum_{b} \sum_{(d,q)\in\mathcal{B}^b} m((d,q),m^b) \tag{3}$$

For a known size B, this defines an asymmetric K-Means clustering problem. A solution can be efficiently computed using extremely fast Euclidean K-Means packages be treating each data point as two separate vectors $\phi(d) \oplus \psi(q)$ and $\psi(q) \oplus \phi(d)$ where $\oplus$ is concatenation.

Cluster Embeddings. Since clustering is performed before training, we do not have dense encoders for ϕ and ψ when constructing the groups. Borrowing methods from hard-negative mining (Robinson et al., 2021) we can replace the ϕ and ψ with a simpler embedding model when constructing groups. We experiment with a sparse vector representation and with pretrained dense representations, settling on GTR (Ni et al., 2021), a popular and generic text embedding model.

Filtering False Negatives. Our method is especially sensitive to false negatives, as they will be more likely to be included in a given batch. Unfortunately, traditional retrieval datasets are not designed with this type of global objective in mind: false negatives are common in most retrieval datasets and their prevalence increases with dataset scale. As one datapoint, Qu et al. (2021) found that over 70% of top-retrieved passages in MS Marco are false negatives.

To avoid a situation where each batch contains a large number of false negatives, we compute an equivalence class: $S(q,d) = \{d' \in \mathcal{D} \mid f(q,d') \geq f(q,d)\}$ for some surrogate scoring function f. At training time, we alter the partition function for d so that it no longer includes the elements of $S(q,d)$, which are not definitively negative examples:

$$\log p(d \mid q) = \frac{\exp f(d,q)}{\exp f(d,q) + \sum_{d' \notin S(q,d)} \exp f(d',q)} \tag{4}$$

For simplicity, we again select f to be a simple pre-trained embedding model. This method likely over-prunes some potential true negatives found by the surrogate model; however we found it to be critical to model accuracy.

Packing. Clusters found by our algorithm will be of varying sizes, and need to be packed into equal-sized batches. We apply a post-hoc procedure. We consider both random partitioning and grouping via greedy cluster-level traveling salesman, similar to Shi et al. (2024). In both cases, we split large group into into smaller batches, and merge close small batches from within the same domain into evenly-sized batches. This has an added benefit of introducing randomness into the groups when training for multiple epochs. We leave it to future work to analyze the full effects of different packing strategies such as expensive Balanced K-Means or heuristic approaches such as Equal K-Means (Gururangan et al., 2023).

4.2 Contextual Document Embedding (CDE)

Contextualization can also be added directly to the archiecture. Taking inspiration from sparse vector retrieval which uses corpus statistics to determine the form of the embedding, we modify the encoders to have access to the corpus itself, i.e. $\phi(d;\mathcal{D})$ and $\psi(d;\mathcal{D})$. This effectively augments the biencoder model to give it the ability to contextualize documents directly.

The main challenge is how to design a neural architecture that can take into account dataset contextualization. On one extreme, we could follow methods like BM25 and precompute a fixed set of corpus statistics that could be fed to the document encoder. On the other extreme, we could allow the encoder full access to the entire corpus, through some form of cross attention. The latter approach has been explored on a small scale in methods like neural processes (Garnelo et al., 2018); however, it would be difficult to scale to larger datasets.

We opt for a middleground that allows the model to learn corpus statistics, but is also relatively efficient to compute, shown in Figure 1. Specifically, we note that document embeddings retain a surprising amount of lexical information even after embedding (Morris et al., 2023). Therefore, if we pre-embed a subset of the corpus, we believe we can still dynamically calculate key dataset information during encoding.

We produce contextualized embeddings via a two-stage process:

First stage: *Gather and embed context.* Given context documents $d^1, ..., d^J \in \mathcal{D}$, we embed each using a unique embedding model and concatenate embeddings into a sequence $M_1(d^1)...M_1(d^J)$.

Second stage: *Embed document with additional context tokens.* To compute ϕ for document d' we integrate contextual embedding sequence at the input of second-stage embedding model M_2:

$$\phi(d'; \mathcal{D}) = M_2(M_1(d^1), \ldots, M_1(d^J), E(d'_1), \ldots, E(d'_T)) \tag{5}$$

Here M_1 is the first-stage encoder model, M_2 is a second-stage encoder model, and E is the token embedding matrix of M_2 applied to each token in d'. In practice, we parameterize both M_1 and M_2 using traditional bidirectional transformers, so our model is comprised of two biencoder-like backbones called in sequence.

There is a similar contextualized model for the query encoder ψ which is also given document context (as we do not have query context at test time):

$$\phi(q; \mathcal{D}) = M_2(M_1(d^1), \ldots, M_1(d^J), E(q_1), \ldots, E(q_T)) \tag{6}$$

We note several implementation properties of this architecture. During training, computing contextual embeddings for each contextual document for each training instance would naively increase training by a computational factor proportional to J, the number of documents in context. This time increase would not be tractable, since contrastive training can already take many days. We overcome this difficulty by sharing context $d^1, ..., d^J$ within a batch of documents; this allows us to compute representations just once per training step and reuse them between documents via computational graph. [2]

When indexing a new corpus $\mathcal{D}$, first stage representations $M_1(d^1)...M_1(d^J)$ can be computed once and cached, so M_1 does not add parameters or runtime to the search process. Query representations can also use the cached context, which only require additional inputs to the encoder. (Our model does not include contextualized queries, only documents, as we typically do not assume access to example queries at test-time.)

Embedding *without* context. Individual corpora during training may not have sufficient or available context. To improve our model's generalization, we use *sequence dropout*, where we randomly replace context embeddings $M_1(d^*)$ with some null token $v_\emptyset$ according to some a uniform probability p.

At test time, if no corpus information is available, our model can now function as a non-contextual biencoder simply by replacing all sequence token inputs with $v_\emptyset$.

Position-agnostic embedding. Since documents of $\mathcal{D}$ are unordered, we remove all positionality from the neural encodings. When parameterizing θ with a traditional transformer, this can be achieved by omitting positional embeddings at the positions corresponding to $\mathcal{D}$. In practice, we use transformers implementations dependent on FlashAttention with rotary positional embeddings at each self-attention layer. Full details of how we disable positionality are available in Section 9.4.

Two-stage gradient caching. To improve training we employ a gradient-caching technique analogous to a two-stage version of GradCache (Gao et al., 2021). This technique allows us to fit larger batches, longer sequences with more contextual samples without running out of memory. Essentially, we compute first-stage and second-stage representations independently without gradients. We then use these frozen representations to compute the loss, and gradients with respect to the second-stage representations. We then re-run the second stage with gradients enabled and use the output gradients to backpropagate through the second-stage model, and obtain gradients for the first-stage representations. We repeat this process for the first-stage representations. This allows us to tradeoff computation (running each transformer forward pass twice) for memory.

[2]Context reuse is only feasible because documents within the same batch typically share a large amount of context anyway, since they are clustered.

Contextual						
Batch	Arch	Batch Size	Cluster Size	Train loss	Train acc.	NDCG@10
		16384	-	0.39	90.3	59.9
✓		512	512	0.81	77.7	61.7
	✓	16384	-	0.37	90.7	62.4
✓	✓	512	512	0.68	80.9	**63.1**

Table 1: Performance of our small models with and without the two improvements proposed in this paper, measured on a shortened version of the BEIR benchmark. Numbers are NDCG@10.

5 Experimental Setup

We consider a range of retrieval experiments across different scales. To run experiments across a suitable number of settings, we devise a small setting: six-layer transformer, maximum sequence length of 64, and maximum number of 64 additional contextual tokens. In this scenario, we evaluate on a truncated version of the BEIR benchmark (Thakur et al., 2021). Given the low cost of each experiment, we are able to pre-train and fine-tune both biencoder and contextual models across a variety of batch sizes in $\{256, 512, 1024, 2048, 4096\}$ and cluster sizes $\{64, 256, 1024, 4096, ..., 2097152, 4194304\}$. As typical state-of-the-art text embedding models are trained in two phases, a large weakly-supervised pre-training phase and a short supervised phase, we run all experiments for both phases.

For the large setting, we use the best settings found via small experiments. We train a single model on sequences of length 512 with 512 contextual documents, evaluating on the full MTEB benchmark (Muennighoff et al., 2022). This includes tasks from retrieval as well as tasks like classification, clustering, and reranking.

Training Data and Metrics We train on the meta-datasets collected in Nussbaum et al. (2024) for training text embedding models. This collection of datasets includes data from 24 datasets scraped from web sources such as Wikipedia and Reddit. Our unsupervised training phase trains on 200M weakly-supervised datapoints scraped from large internet sources such as Reddit and Wikipedia. The supervised training phase includes 1.8M human-written query-document pairs intended for text retrieval, and is aggregated from popular retrieval datasets such as HotpotQA and MS MARCO (Yang et al., 2018; Bajaj et al., 2018). For our full model, we also consider supervised training on the BGE meta-datasets (Xiao et al., 2024). We evaluate our models using NDCG@10, a conventional retrieval metric that enables comparison across many disparate datasets.

Implementation When partitioning our dataset into batches, we encode documents and queries using GTR (Ni et al., 2021) and implement our clustering algorithm on top of FAISS (Douze et al., 2024). We cluster per-domain for 100 steps and take the best clustering out of 3 attempts. We select NomicBERT as our pre-trained model backbone (Nussbaum et al., 2024), which has 137M parameters. We prepend all texts with short task-specific prefixes to identify each task; prefixes are listed in Section 9.7.

Training We initialize both M_1 and M_2 using the BERT-base model from Nussbaum et al. (2024) that includes flash attention. Weights are shared between ϕ and ψ, but notably not between M_1 and M_2. For all experiments, we train with the Adam optimizer with 1000 steps of warmup to a learning rate of $2 \cdot 10^{-5}$ and linearly decay to 0 throughout training. We train for three epochs unless otherwise specified. We set the maximum sequence length for all inputs to 512 and the number of contextual inputs to 512 (so the second-stage model has an input length of 1024). When computing contrastive loss, we use a fixed temperature of $\tau = 0.02$. When sequence dropout is enabled in our contextual architecture, we set contextual input tokens to null vectors with a uniform probability $p = 0.005$. If the batch size exceeds the number of contextual documents, we randomly sample to produce contextual inputs.

	Clssfctn	Cluster	PairCls	Rerank	Retrvl	STS	Summ.	Mean
nomic-embed-v1	74.1	43.9	85.2	55.7	52.8	82.1	30.1	62.39
stella-base-en-v2	75.3	44.9	86.5	58.8	50.1	83.0	32.5	62.61
bge-base-en-v1.5	75.5	45.8	86.6	58.9	53.3	82.4	31.1	63.56
GIST-Embedding-v0	76.0	46.2	86.3	59.4	52.3	83.5	30.9	63.71
gte-base-en-v1.5	77.2	46.8	85.3	57.7	54.1	82.0	31.2	64.11
`anon-model-v1`								
[Random]	81.3	46.6	84.1	55.3	51.1	81.4	31.6	63.81
[Contextual]	81.7	48.3	84.7	56.7	53.3	81.6	31.2	**65.00**

Table 2: Performance of models with 250M or fewer parameters on the MTEB benchmark for text embedding models. “Random” indicates the performance of our model with random training documents included instead of per-domain contextual documents.

6 Results

The main results are highlighted in Table 1 and Section 6. In the smaller setting, we observe that both adversarial contrastive learning and our contextual architecture improve performance compared to vanilla biencoder training. We observe the largest improvement when we combine these techniques.

Contextual batching After controlling for batch size and filtering for false negatives, we observe a strong correlation (visualized in Figure 2) between batch difficulty and downstream performance: *reordering datapoints to make batches harder definitively enhances overall learning*. This corroborates prior findings (Xiong et al., 2020; Qu et al., 2021) and theory (Zhang & Stratos, 2021) that more difficult batches in contrastive learning form a better overall gradient approximation and learn more effectively.

Section 6 showcases model performance across batch and cluster sizes after both phases of training. We observe that although a large batch and cluster size are useful when filtering is not enacted, when including filtering, smaller cluster (and harder) are clearly better, and large batches do not add much. When comparing filtered to non-filtered models (Figure 4), filtering false negatives clearly improves performance.

Contextual architecture In addition to adversarial batching, we compare our contextual architecture to a biencoder across the datasets of BEIR in Table 1. Our architecture generally matches or improves performance on all downstream datasets, with largest improvements in ArguAna and SciFact, two of the smaller and more out-of-domain datasets.

Full-scale training Figure 5 shows our models' performance when trained for multiple epochs on the supervised datasets, relative to the best similar-sized embedding model (dashed line). We find best performance when training for four epochs on the BGE meta-datasets. Although our best model does use a single hard negative per query, we are still able to to achieve state-of-the-art performance without using *any* hard negatives.

For our final model (`anon-model-v1`), we select the best of the supervised models, which comes from finetuning on the BGE dataset. On MTEB, `anon-model-v1` obtains state-of-the-art results compared to models of the same size. Although inspired by problems in the specific domain of text retrieval, we observe that our approach improves embedding performance in all domains, including clustering, classification, and semantic similarity. We also evaluate a “random documents” baseline, where we sample random documents from the training dataset to simulate a scenario where we lack access to the test corpus. In this setting, we drop around 1.2 points on average across all tasks; the STS tasks in particular appear to produce representations that are close to context-agnostic.

7 Analysis

How hard are our clusters? To analysis the relationship between cluster size in our clustering algorithm and the overall average difficulty of in-batch negatives, we measure the average difficulty

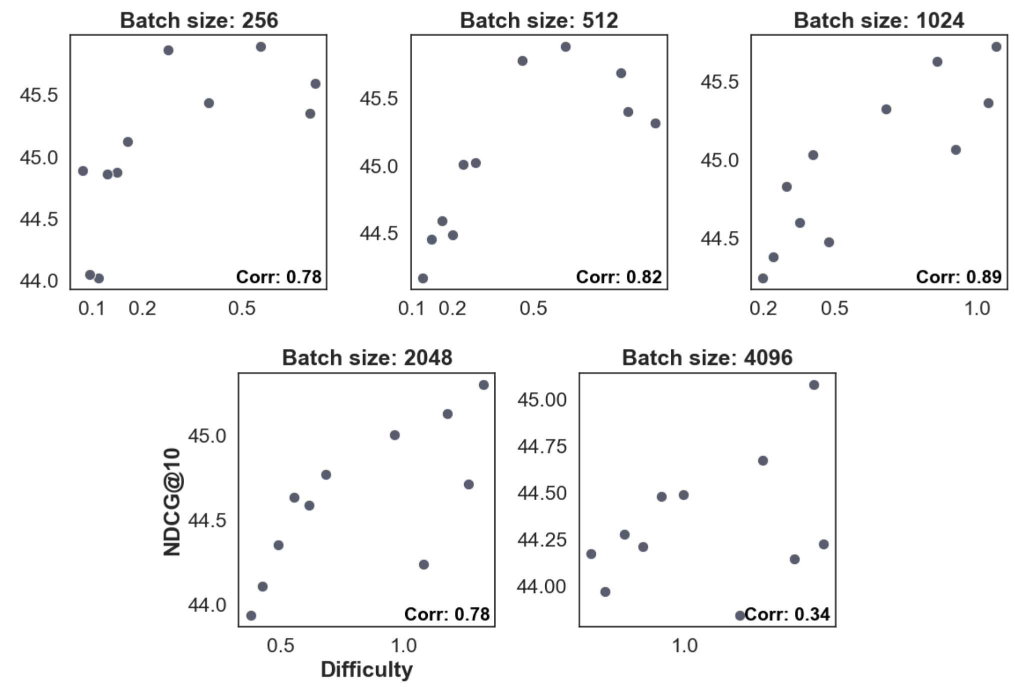

Figure 2: Performance vs. average batch difficulty (as measured by loss at the end of pre-training and supervised training) across batch sizes, after supervised contrastive training. Within a given batch size, we observe a clear increase in performance by making individual batches harder. Correlations are Pearson.

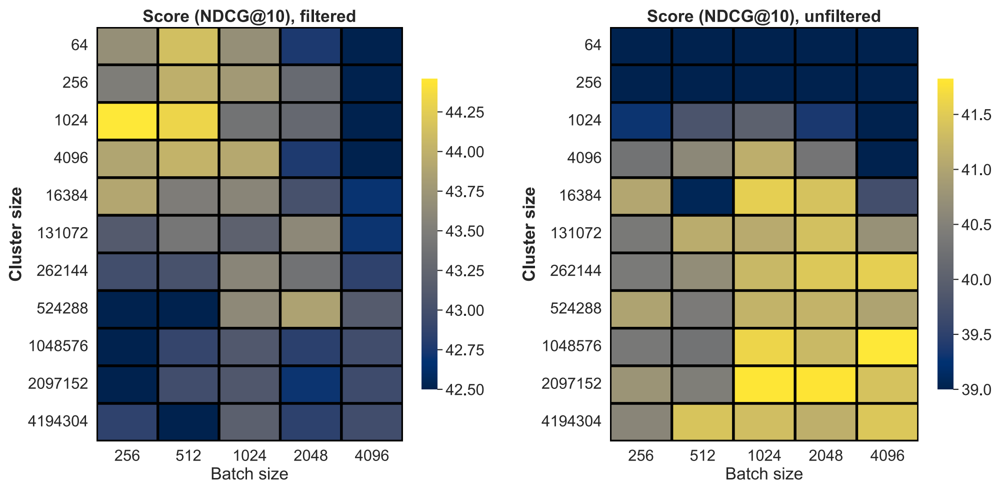

Figure 3: Biencoder performance with filtering (left) and without (right) across batch and cluster sizes during unsupervised contrastive pre-training. With filtering, small cluster sizes clearly improve performance, and larger batch sizes do not.

of 1000 batches across a variety of batch and cluster sizes and plot the data in Figure 6. We observe that larger batches bring easier non-negative examples, and decreasing cluster size clearly increases the average hardness of negative examples in a given cluster.

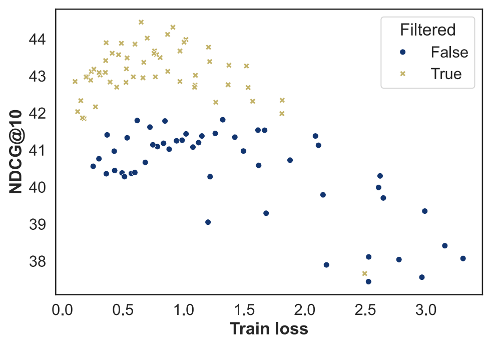

Figure 4: Impact of filtering during training across various batch and cluster sizes. Each dot is a biencoder pretrained with a different batch and cluster size.

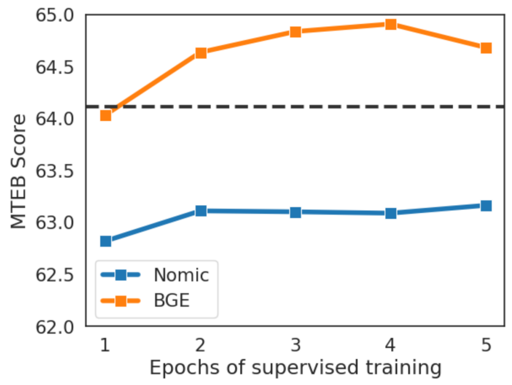

Figure 5: Performance on MTEB across epochs of supervised training on the Nomic and BGE supervised meta-datasets.

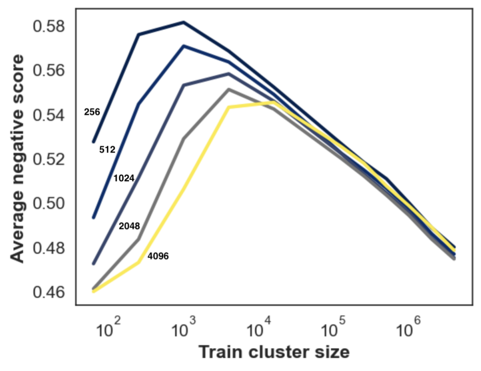

Figure 6: Average difficulty of in-batch negatives as measured by a surrogate model as cluster size and batch size change.

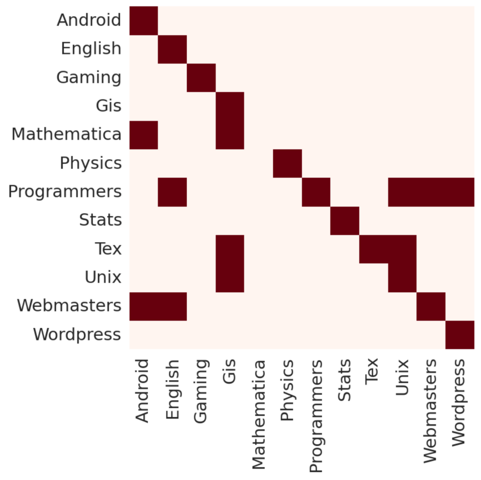

Figure 7: Impact of context by testing our model with different Stackexchange forum input types. Y-axis indicates the input domain, X-axis indicates the test domain. Dark squares come within one point NDCG@10.

Which contextual documents help? To confirm that the CDE model is utilizing contextual information from $\mathcal{D}$ we consider how different contextual documents help for a given docuent d. Figure 7 measures results on CQADupstack, a collection of Stack Exchange forum posts. We randomly sample inputs to from $\mathcal{D}$ from a domain (x-axis) and use them as input to the downstream task d marked along the y-axis. We mark a square as red if its score comes within 1 point of NDCG of the top score for its domain. Generally utilizing the documents for that domain are best, but there are some cross-over interactions. Full results for each input-output task pair are shown in the appendix.

8 Conclusion

We propose two improvements to traditional biencoder models for generating embeddings. The first improvement involves an algorithm for reordering training datapoints to make batches harder and improves vanilla training with minimal changes. Our second improvement involves a new corpus-aware architecture for retrieval and allows us to train a state-of-the-art text embedding model.

REFERENCES

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset, 2018. URL https://arxiv.org/abs/1611.09268.

Sebastian Borgeaud, Arthur Mensch, Jordan Hoffmann, Trevor Cai, Eliza Rutherford, Katie Millican, George van den Driessche, Jean-Baptiste Lespiau, Bogdan Damoc, Aidan Clark, Diego de Las Casas, Aurelia Guy, Jacob Menick, Roman Ring, Tom Hennigan, Saffron Huang, Loren Maggiore, Chris Jones, Albin Cassirer, Andy Brock, Michela Paganini, Geoffrey Irving, Oriol Vinyals, Simon Osindero, Karen Simonyan, Jack W. Rae, Erich Elsen, and Laurent Sifre. Improving language models by retrieving from trillions of tokens, 2022.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations, 2020.

William Coster and David Kauchak. Simple English Wikipedia: A new text simplification task. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea (eds.), *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pp. 665–669, Portland, Oregon, USA, June 2011. Association for Computational Linguistics. URL https://aclanthology.org/P11-2117.

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, 2022.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library, 2024.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Open Question Answering Over Curated and Extracted Knowledge Bases. In *KDD*, 2014.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: long form question answering. In Anna Korhonen, David R. Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pp. 3558–3567. Association for Computational Linguistics, 2019. doi: 10.18653/v1/p19-1346. URL https://doi.org/10.18653/v1/p19-1346.

Katja Filippova and Yasemin Altun. Overcoming the lack of parallel data in sentence compression. In David Yarowsky, Timothy Baldwin, Anna Korhonen, Karen Livescu, and Steven Bethard (eds.), *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pp. 1481–1491, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://aclanthology.org/D13-1155.

Wikimedia Foundation. Wikimedia downloads, 2024. URL https://dumps.wikimedia.org.

Luyu Gao and Jamie Callan. Unsupervised corpus aware language model pre-training for dense passage retrieval, 2021.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup, 2021.

Marta Garnelo, Dan Rosenbaum, Chris J. Maddison, Tiago Ramalho, David Saxton, Murray Shanahan, Yee Whye Teh, Danilo J. Rezende, and S. M. Ali Eslami. Conditional neural processes, 2018.

Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C Lipton. Amazonqa: A review-based question answering task, 2019.

Suchin Gururangan, Margaret Li, Mike Lewis, Weijia Shi, Tim Althoff, Noah A. Smith, and Luke Zettlemoyer. Scaling expert language models with unsupervised domain discovery, 2023.

Felix Hamborg, Norman Meuschke, Corinna Breitinger, and Bela Gipp. news-please: A generic news crawler and extractor. In *Proceedings of the 15th International Symposium of Information Science*, pp. 218–223, March 2017. doi: 10.5281/zenodo.4120316.

Christopher Hidey and Kathy McKeown. Identifying causal relations using parallel Wikipedia articles. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1424–1433, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1135. URL https://aclanthology.org/P16-1135.

Hamel Husain, Ho-Hsiang Wu, Tiferet Gazit, Miltiadis Allamanis, and Marc Brockschmidt. CodeSearchNet challenge: Evaluating the state of semantic code search. *arXiv preprint arXiv:1909.09436*, 2019.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. Unsupervised dense information retrieval with contrastive learning, 2022.

Minguk Jang, Sae-Young Chung, and Hye Won Chung. Test-time adaptation via self-training with nearest neighbor information, 2023.

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, 2020.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. Generalization through memorization: Nearest neighbor language models, 2020.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. Gooaq: Open question answering with diverse answer types, 2021.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020. URL https://arxiv.org/abs/2004.12832.

Hyunjik Kim, Andriy Mnih, Jonathan Schwarz, Marta Garnelo, Ali Eslami, Dan Rosenbaum, Oriol Vinyals, and Yee Whye Teh. Attentive neural processes, 2019.

Jannik Kossen, Neil Band, Clare Lyle, Aidan N. Gomez, Tom Rainforth, and Yarin Gal. Self-attention between datapoints: Going beyond individual input-output pairs in deep learning, 2022.

Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset, 2018.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them, 2021.

Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 4482–4491, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1478. URL https://aclanthology.org/D18-1478.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. Towards general text embeddings with multi-stage contrastive learning, 2023.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S. Weld. S2orc: The semantic scholar open research corpus, 2020.

Jiawei Ma, Po-Yao Huang, Saining Xie, Shang-Wen Li, Luke Zettlemoyer, Shih-Fu Chang, Wen-Tau Yih, and Hu Xu. Mode: Clip data experts via clustering, 2024.

John X. Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M. Rush. Text embeddings reveal (almost) as much as text, 2023.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*, 2022. doi: 10.48550/ARXIV.2210.07316. URL https://arxiv.org/abs/2210.07316.

Tung Nguyen and Aditya Grover. Transformer neural processes: Uncertainty-aware meta learning via sequence modeling, 2023.

Jianmo Ni, Jiacheng Li, and Julian McAuley. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan (eds.), *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pp. 188–197, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1018. URL https://aclanthology.org/D19-1018.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. Large dual encoders are generalizable retrievers, 2021.

Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert, 2020.

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2024.

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, 2021.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ Questions for Machine Comprehension of Text. *arXiv e-prints*, art. arXiv:1606.05250, 2016.

Nils Reimers, Elliot Choi, Amr Kayid, Alekhya Nandula, Manoj Govindassamy, and Abdullah Elkady. Introducing embed v3, Nov 2023. URL https://txt.cohere.com/introducing-embed-v3/.

Stephen Robertson and Hugo Zaragoza. *The Probabilistic Relevance Framework: BM25 and Beyond*. Now Publishers Inc., 2009.

Joshua Robinson, Ching-Yao Chuang, Suvrit Sra, and Stefanie Jegelka. Contrastive learning with hard negative samples, 2021.

J. J. Rocchio. Relevance feedback in information retrieval. 1971. URL https://api.semanticscholar.org/CorpusID:61859400.

Vin Sachidananda, Ziyi Yang, and Chenguang Zhu. Global selection of contrastive batches via optimization on sample permutations. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 29542–29562. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/sachidananda23a.html.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. Colbertv2: Effective and efficient retrieval via lightweight late interaction, 2022. URL https://arxiv.org/abs/2112.01488.

Christopher Sciavolino. Towards universal dense retrieval for open-domain question answering, 2021.

Abigail See, Peter J. Liu, and Christopher D. Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1073–1083, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1099. URL https://www.aclweb.org/anthology/P17-1099.

Weijia Shi, Sewon Min, Maria Lomeli, Chunting Zhou, Margaret Li, Gergely Szilvasy, Rich James, Xi Victoria Lin, Noah A. Smith, Luke Zettlemoyer, Scott Yih, and Mike Lewis. In-context pretraining: Language modeling beyond document boundaries, 2024.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. One embedder, any task: Instruction-finetuned text embeddings, 2023.

Mujeen Sung, Jungsoo Park, Jaewoo Kang, Danqi Chen, and Jinhyuk Lee. Optimizing test-time query representations for dense retrieval, 2023.

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

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Simlm: Pre-training with representation bottleneck for dense passage retrieval, 2023.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. Text embeddings by weakly-supervised contrastive pre-training, 2024.

Xiao Wang, Craig Macdonald, Nicola Tonellotto, and Iadh Ounis. Pseudo-relevance feedback for multiple representation dense retrieval. In *Proceedings of the 2021 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '21. ACM, July 2021. doi: 10.1145/3471158.3472250. URL http://dx.doi.org/10.1145/3471158.3472250.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packaged resources to advance general chinese embedding, 2024. URL https://arxiv.org/abs/2309.07597.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020. URL https://arxiv.org/abs/2007.00808.

Canwen Xu, Daya Guo, Nan Duan, and Julian McAuley. Laprador: Unsupervised pretrained dense retriever for zero-shot text retrieval, 2022.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018. URL https://arxiv.org/abs/1809.09600.

Wenzheng Zhang and Karl Stratos. Understanding hard negatives in noise contrastive estimation, 2021.