Knowledge Distillation of Text Embedding Models with Teacher-Aligned Representations

Anonymous ACL submission

Abstract

We present REDACT, a knowledge distillation framework for text embedding models. A key distinguishing feature is that our distilled `redact` models are compatible with their teacher, enabling flexible asymmetric architectures where documents are encoded with the larger teacher model, while queries use smaller `redact` models. We also show that `redact` models automatically inherit MRL and robustness to output quantization whenever these properties are present in the teacher model, without explicitly training for them. To demonstrate the effectiveness of our framework we publish `redact-ir`, a 23M parameters information retrieval oriented model that, besides being teacher-compatibile, sets a new state-of-the-art (SOTA) on BEIR, ranking no.1 on the public leaderboard for models of its size. Asymmetric mode further increases its retrieval performance. Our scheme is however not restricted to information retrieval. We demonstrate its wider applicability by synthesizing the multi-task `redact-mt` model. This also sets a new SOTA, achieving no.1 on the public MTEB v2 (English) leaderboard for models of its size. REDACT is applicable to black-box models, requires no judgments nor hard negatives, and training can be conducted using small batch sizes. Thus, dataset and training infrastructure requirements for our framework are modest. We make our models publicly available under a permissive Apache 2.0 license.

1 Introduction

Recent advancements in neural networks have paved the way for drastic improvements in a wide array of natural language processing tasks, and new use cases like retrieval augmented generation ("RAG") are fueling a massive increase in user demand for these Transformer-based (Vaswani et al., 2017) models.

Unfortunately, this dramatic performance jump has come at the cost of an explosion in the size of these models, resulting in massive costs in terms of hardware, electric power, and maintenance required to train and operate them. To accommodate varying budgets, model providers typically offer a variety of model sizes. In the case of bi-encoder text embedding models, however, none of these models, even different sizes from the same model family, are compatible with each other to the best of our knowledge. As a result, users desiring to switch to a different embedding model are required to perform a costly complete recomputation of the embeddings of all their data. In the context of information retrieval (IR), this incompatibility also leads to rigid architectures where the same large model needs to be utilized to encode both documents as well as queries. But the system requirements of these two phases of IR are not symmetric: while documents are typically embedded only once at index time and under mild latency requirements, embedding user queries using large models can cause substantial sustained operational costs and can incur prohibitive latencies in end-user experience.

To address these challenges we propose REDACT, a knowledge distillation *framework* consisting of model architecture, training loss, training regime and training datasets. REDACT produces text embedding models that are *aligned* to their teacher. The compatibility of `redact` models enables flexible architectures like the one shown in Figure 2 where document embeddings are computed using large and expensive models, while queries can be more economically encoded with the smaller `redact` models.

We demonstrate the efficacy of our framework by first synthesizing an information retrieval oriented model called `redact-ir`. This 23M parameter model is a distillation of a 109M teacher (4.7× compression), leading to a 6.5× and 7.3× through-

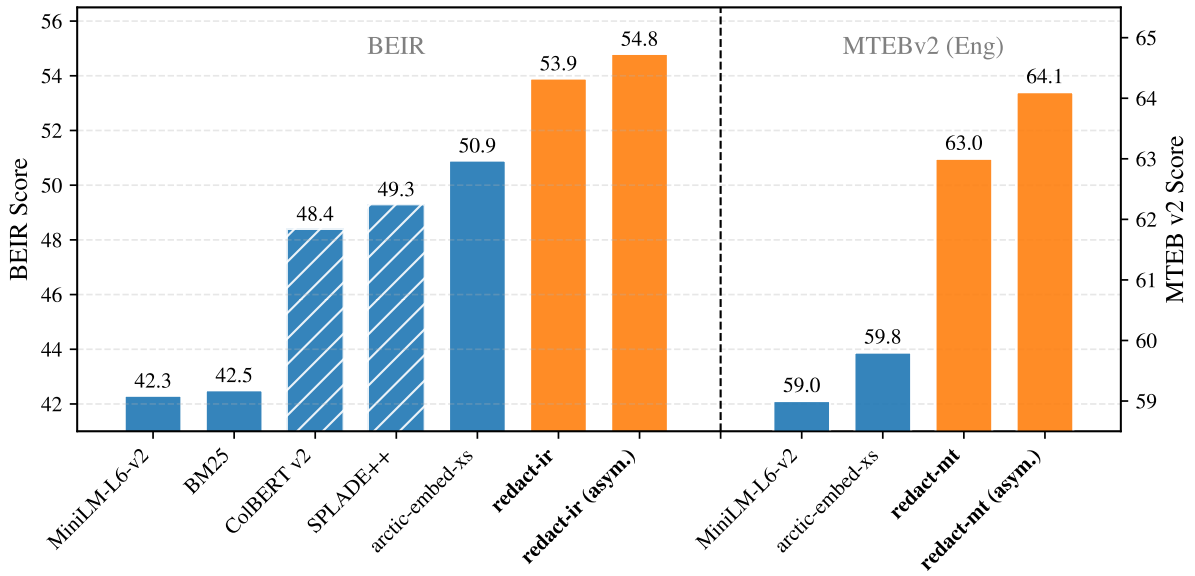


Figure 1: Our information retrieval oriented `redact-ir` model (23M parameters) sets a new state-of-the-art on the BEIR benchmark for $\leq$100M parameters models. When run in asymmetric mode, its retrieval performance is further increased. Our multi-task `redact-mt` model (23M parameters) also sets a new state-of-the-art on MTEB v2 (English). Hatched columns indicate comparison models that leverage larger (110M parameters) models.

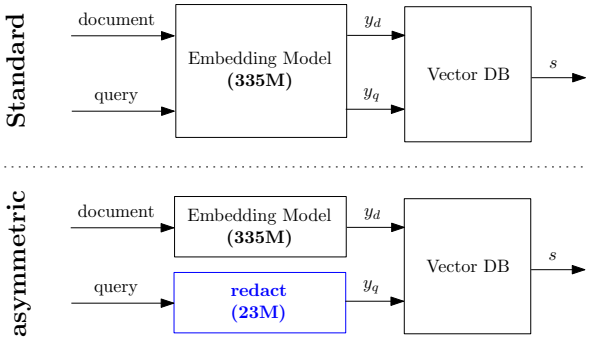


Figure 2: REDACT-enabled asymmetric architecture.

put increase when encoding documents and queries respectively. When run in standard mode, encoding both queries and documents, `redact-ir` sets a new SOTA BEIR (Thakur et al., 2021) score for models of its size, ranking no.1 on the corresponding public leaderboard at the time of writing.

One of our core contributions is to show that taking advantage of the asymmetric system in Figure 2 unlocks further effectiveness gains. Namely, Figure 1 illustrates that we set even higher SOTA scores in settings where query time has sufficient computational budget only to run the student model, while documents can be processed with the larger teacher. We generally find that IR performance of the asymmetric system is somewhere in the middle between running the teacher and the student models in their respective standard modes.

Our scheme is not specialized to IR. To demonstrate REDACT's broader applicability we synthesize the multi-task `redact-mt` model, by performing an even more aggressive distillation of 335M parameters down to 23M. This leads to throughput gains by a factor of 24.4× and 23.7× for documents and queries respectively. We assess its performance on MTEB v2-English (Enevoldsen et al., 2025), a benchmark that, besides IR, tests models' performance on other tasks such as classification, clustering, reranking, semantic sentence similarity, and summarization. When run in standard mode, `redact-mt` also sets a new SOTA on this benchmark and ranks no.1 on the corresponding public leaderboard for models of its size. Taking advantage of asymmetric mode further increases performance also in this case, as shown on Figure 1.

We furthermore show that `redact` models automatically inherit Matryoshka Representation Learning (MRL, Kusupati et al., 2022) and robustness to quantization properties whenever the teacher model has them, without specifically training for them. MRL allows the truncation of embedding vectors at arbitrary lengths, while vector quantization allows the storage of embedding vectors using more compact types, i.e., `int8` or `binary` instead of `float32`. Both of these techniques aim at reducing storage space and increasing retrieval speed, by gradually trading off retrieval performance.

REDACT can be applied to black-box models since, in contrast to several other knowledge distillation frameworks (Wang et al., 2020; Sanh et al., 2020; Jiao et al., 2020; Sun et al., 2020), it does not require access to any of the model's internals such as keys, queries and values. Further, it can be applied in cases where the architectures of redact and its teacher are different: they can, for instance, use different tokenizers, have a different number of layers and attention heads, or different hidden state dimensions.

A second advantage of REDACT is that it is not based on contrastive loss and hence does not require the availability of judgments or hard negatives as training datasets. These are typically needed to train embedding models (Neelakantan et al., 2022; Formal et al., 2021; Santhanam et al., 2022) as well as some other knowledge distillation procedures designed for text embedding models (Hofstätter et al., 2020).

Third, as elaborated in Section 3.2, training works well also with small batch sizes. As a result, we were able to train the two aforementioned SOTA models on a single A100 GPU within a budget of 100 hours each.

We describe REDACT as a *lightweight* knowledge distillation scheme, thanks to the deliberate simplicity of the loss, entailing exclusively the ℓ_2 norm of the approximation error between student and teacher representations, and model architecture, along with simplified training dataset and training infrastructure requirements. A key contribution of our work is to show that such an uncomplicated setup is sufficient to derive SOTA text embedding models.

Finally, we reflect on the system's robustness to perturbations. For instance, we expect documents with similar meanings but different formulations to score similarly against a given query. We consider our work as seeking to synthesize models that directly approximate the function that maps strings to embeddings of their teacher. In Section 4.3 we find that downstream task performance is similarly robust to the perturbations introduced by our approximation scheme. We observe that a substantial amount of approximation error can be absorbed by the system and call this its *robustness margin*.

2 Related Work

Despite the simplicity of the approach proposed in this paper, literature explicitly exploring its potential remains sparse.

Closest to our work is (Campos et al., 2023), which studies asymmetric retrieval supported by query and document encoders of different sizes. Distillation of the query encoder is achieved via layer pruning of a Transformer backbone pretrained to be compatible with the document encoder, followed by alignment using a KL divergence loss. However, this work does not aim to synthesize competitive embedding models, but rather to analyze the impact of pruning on retrieval performance. Further, we allow independently trained query and document encoders and do not assume access to teacher weights.

The work in (Yoon and Arik, 2025), although not aimed at solving the query/document asymmetry problem, trains an MLP converter network to translate embeddings between models. Unlike our findings, they report that regression alone is insufficient and consequently introduce additional loss components. In their setup, both Transformer models are frozen and only the MLP is trained, which we believe explains the differing conclusions.

The recipe in (Hugging Face, 2024) down-scales teacher embeddings to the student's dimensionality using PCA and aligns them via MSE loss. However, PCA destroys properties such as MRL and robustness to quantization. Moreover, since we propose to use the teacher for document encoding and optionally a student for queries, dimensionality reduction degrades document representations regardless of whether the student is deployed. Our approach avoids this issue by training students directly in the teacher's embedding space.

MSE loss has also been used in (Reimers and Gurevych, 2020) to extend monolingual models to multilingual settings, although in this case the student is typically larger than the teacher.

Several prior works propose distillation schemes for Transformer-based language models, including TinyBERT (Jiao et al., 2020; Chen et al., 2021), DistilBERT (Sanh et al., 2020), and MobileBERT (Sun et al., 2020). These methods target language modeling heads and are not directly applicable to embedding models. MiniLM (Wang et al., 2020) is applicable to embeddings but requires access to model internals, matching tokenizers, and the same number of attention heads. (Truong et al., 2025) proposes a solution to the problem of distilling models with different tokenizers using optimal transport ideas. The approach in (Hofstätter et al., 2020) introduces a Margin MSE loss, later adopted

by SPLADE v2 (Formal et al., 2021), but relies on judgments and hard negatives, unlike our method.

3 Approach

The architecture used in this work is shown in Figure 3. It consists of a Transformer $\mathrm{Backbone}$, followed by mean pooling. In order to match the target teacher model's output dimension d we stack $W^{\text{out}} \in \mathbb{R}^{d' \times d}$, a Linear layer that maps the Backbone's (typically lower) dimensions d' into d.

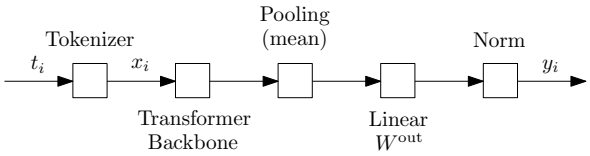


Figure 3: `redact` model architecture.

Let I be an index over the training set, and $\{(t_i, \hat{y}_i)\}_{i \in I}$ be the collection of tuples of training texts t_i (strings) and their corresponding ground truth embeddings $\hat{y}_i \in \mathbb{R}^d$ produced by the teacher model. The texts t_i can entail both queries as well as documents. Let $x_i \in \mathbb{N}^T$ be the tokenization of t_i, where T is the sequence length. Padding tokens `[PAD]` are appended to make the lengths of sequences in a batch match. Respectively, sequences are truncated if they exceed the model's maximum context length $\overline{T}$.

Then, the embedding vector $y_i \in \mathbb{R}^d$ computed by our model for a given input sequence x_i is given by

$$y_i = \mathrm{Norm}(\mathrm{Linear}(\mathrm{Mean}(\mathrm{Transf.}(x_i)))), \quad (1)$$

where mean pooling Mean is computed only over Transformer outputs corresponding to non-`[PAD]` tokens. We use mean pooling independently of the type of pooling utilized by the teacher model. We have experimented with `[CLS]`-token pooling, but found it to be an inferior choice, see Appendix A.1. The normalization layer $\mathrm{Norm}(\cdot)$ is added to the stack if the teacher model is structured to produce normalized embedding vectors, i.e.,

$$\|\hat{y}_i\|_2 = 1 \quad \forall i \in I.$$

In this work, we intend to train the model given by Eq. (1) so that y_i approximates $\hat{y}_i$. Accordingly, we define the approximation error

$$e_i = y_i - \hat{y}_i, \quad (2)$$

and utilize it as a knowledge distillation training loss:

$$\mathcal{L}_i = \|e_i\|_2. \quad (3)$$

We propose to limit the training loss to the ℓ_2 expression in Eq. (3). This choice enables the distillation procedure presented in this paper to be applicable also in cases when no internals of the models, such as keys, queries and values, are known. That is, it can be applied to any black-box model for which the training tuples $(t_i, \hat{y}_i)$ can be obtained. Note also that, as mentioned in the Introduction, the computation of this loss does not require judgments nor hard negatives, and that it is also not specific to the information retrieval (IR) task.

Appendix B reports training loss alternatives to Eq. (3) based on other knowledge distillation approaches available in the literature. Specifically, we extended Eq. (3) with the losses proposed by TinyBERT (Jiao et al., 2020), DistilBERT (Sanh et al., 2020) and MiniLM (Wang et al., 2020). These, however, did not produce better results in our experiments, and did not have all the benefits enumerated above, so we did not pursue them further. It should also be noted that, as shown in the Appendix, these distillation procedures require the tokenizers to match, and in some cases they also require additional internals, such as the number of attention heads in the Transformer layers, to be the same. Our procedure does not have these restrictions.

3.1 Training Datasets

We selected our training datasets to cover a broad range of topics, including general knowledge, news, science, entertainment, and commerce. Among them, Vocabulary is a new dataset that we created in the context of this work and that we are making publicly available. Vocabulary was synthesized by taking a list of 479k words appearing in English language contexts from (Kaggle, 2023) and prompting Claude 3.5 Sonnet[1] to produce definitions or important facts about them. Appendix C reports the prompt used as well as samples from this dataset.

All the other datasets are also publicly available. We do not perform any processing of the raw data contained in them; we merely extract the relevant column and store it as a parquet file in our processing pipelines.

Our training data consists of both document and query segments. For documents, we utilize 3M

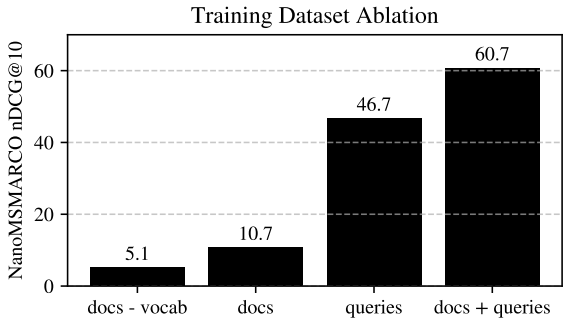
[1] https://www.anthropic.com/

samples from FineWeb (Penedo et al., 2024) (sampled from the 10B tokens subsample), 900k samples from CC-News (100k random documents for each year between 2016 and 2024), 30k BERT tokens (representing each individual token from BERT's tokenizer Devlin et al., 2019 to allow for baseline alignment), and 800k entries from our Vocabulary dataset. For queries, we use 979k samples from Amazon QA (Gupta et al., 2019), 27k from `LoTTE/pooled/` (Santhanam et al., 2022), 502k from `MSMARCO/train/` (Nguyen et al., 2016), 272k from `PubMedQA/pqa_artificial/` and `PubMedQA/pqa_unlabeled/` (Jin et al., 2019), and 73k from Trivia QA (Joshi et al., 2017).

We stress the fact that although some of these datasets also entail judgments, we do not utilize them for training since they are not needed in the computation of the loss in Eq. (3).

Figure 4 shows the relative impact of these training datasets on downstream performance of `redact-ir` on NanoMSMARCO, after training for 1 epoch. These results indicate that training on queries is more important than documents, but to reach state-of-the-art performance it is necessary to train on both.

Figure 4: Model performance when trained for one epoch with different sub-segments of training datasets.



We finally note that we have reviewed the sources of each one of these datasets to avoid contamination with the datasets utilized for the benchmarking results discussed in Section 4. In particular, we consider MSMARCO and its smaller variant NanoMSMARCO as training/dev datasets, as is common in the literature, and do not include them in any of our performance benchmarks.

3.2 Model Training

The Transformer backbone used in this work is `MiniLM-L6-v2` (Wang et al., 2020), a 23M parameter encoder model. We train the parameters of the whole model, which includes the Transformer backbone and W^{out}. We use AdamW as the optimizer, setting the initial learning rate to 1e-4 and leave the other parameters at their defaults ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay $= 0.01$). We train our model for 3 cycles of linear learning rate decay over 10 epochs, resulting in a total of 30 training epochs. The learning rate is linearly decayed to 1e-5 over the 10 epochs of each cycle and then reset to 1e-4 before the following cycle. Appendix A.2 reports our experimentation comparing different learning rate schedules. We found linear decay to work best.

We set the batch size to 32. As shown in (Chen et al., 2020), approaches that rely on contrastive loss typically benefit from larger batch sizes. For instance, (Muennighoff et al., 2025) uses a batch size of 2,048 while in (Yu et al., 2024) the batch size is set to 32,768. In our work we found that a batch size of 32 with our loss in Eq. (3) is sufficient. In fact, the analysis in Appendix A.3 shows that our training regime favors more steps with smaller batch sizes, rather than the other way around. We believe this is owed to the ℓ_2 loss in Eq. (3) providing dense supervision across all embedding dimensions, unlike contrastive approaches that only provide relative ordering supervision between positive and negative pairs.

We precompute and cache the target embeddings $\hat{y}_i$ for all the training samples in the datasets discussed in Sec. 3.1. This substantially decreases training time as we only need to perform forward passes on the student `redact` model. Whenever a model supports instruction prompts (Su et al., 2023), we supply them before computing the corresponding ground truth embedding $\hat{y}_i$, and we also supply them to `redact` when we compute y_i according to Eq. (1).

Overall, we obtain approximately 200k training batches from the datasets described in Section 3.1, out of which we hold out a randomly sampled validation set of 128 batches. The remaining training batches are shuffled before each epoch. With this setup, we train on a single NVIDIA A100 40GB GPU with a budget of 100 hours.

4 Results

4.1 Information Retrieval Model

`redact-ir` is teacher-compatible and sets a new state-of-the-art for compact information retrieval oriented embedding models, while sup-

porting MRL and vector quantization.

We trained `redact-ir`, a model focused on information retrieval (IR). We make this model publicly available under a permissive Apache 2.0 license.

We assess its effectiveness on BEIR (Thakur et al., 2021), an industry standard benchmark for IR systems. Table 1 reports our results on this benchmark, and Figure 1 displays them graphically. We compare against arctic-embed-xs (Merrick et al., 2024), the current state-of-the-art for ≤30M parameters models, as well as other popular retrieval models, including SPLADE++ (Formal et al., 2021) and ColBERT v2 (Santhanam et al., 2022), although we note that these leverage larger (110M parameters) BERT models, and are thus hatched in Figure 1. We also include baseline BM25 (Xhluca, 2023) scores. The last row reports the scores of arctic-embed-m-v1.5 (Merrick et al., 2024), the teacher model `redact-ir` was distilled from. This teacher model was chosen for its strong IR performance at its size.

Scores in the table are bold and underlined when our models improve upon the best comparison methods. We further highlight in blue color scores for which asymmetric mode is an improvement over standard mode.

Our model `redact-ir` ranks no.1 on the public leaderboard[2] of BEIR for models with ≤100M parameters at the time of writing, setting a new state-of-the-art for models of this size. As shown in the table, we set a new state-of-the-art on 9/14 datasets when `redact-ir` is run in standard mode. Our overall average score also sets a new state-of-the-art at an nDCG@10 of 53.9, retaining 96.1% of the teacher's IR performance with 4.7× fewer parameters. On tests run on an AWS EC2 `i3.large` instance, which is a commonly used CPU-only VM for database and search workloads, this leads to inference time speed-ups of 6.5× / 7.3× for docs and queries respectively; details are in Appendix G.

In asymmetric mode, retrieval performance is further increased in 11/14 datasets compared to `redact-ir` standard, as highlighted in blue in the table. It achieves an aggregated score of 54.8 nDCG@10, i.e., it retains 97.7% of the teacher's performance while running a 4.7× smaller model at query time.

These results demonstrate that despite its implementation simplicity and low training and infrastructure requirements, and under a substantial compression regime, `REDACT` is able to synthesize a general purpose SOTA retrieval embedding model that is aligned to its teacher. These results also indicate that `REDACT` is a better approach to synthesizing smaller variants of a model family than training via contrastive loss. In other words, larger models are better suited to optimize the structure of embedding spaces during contrastive training, while smaller models more easily approximate these pre-optimized structures than build them independently. This is evidenced by the fact that we substantially outperform arctic-embed-xs, another 23M parameters model trained using the same procedure and datasets as arctic-embed-m-v1.5, the teacher of `redact-ir`.

Figure 5 shows that `redact-ir` also inherits MRL (Kusupati et al., 2022) and vector quantization properties from the teacher model, without specifically training for them.

4.2 Multi-Task Model

`redact-mt` is teacher-compatible and sets a new state-of-the-art for compact multi-task text embedding models, while supporting MRL and vector quantization.

The training recipe detailed in Section 3 is not specific to information retrieval. We thus trained a 23M parameter multi-task `redact-mt` model, distilling from mxbai-l-v1 (335M parameters). The compression ratio in this case is 14.6×. This model is also publicly available under a permissive Apache 2.0 license.

We assess its performance on MTEB v2-English (Enevoldsen et al., 2025), a multi-task benchmark that, besides retrieval and reranking, also measures classification, clustering, pair classification, semantic textual similarity, and summarization performance.

Table 2 summarizes the results for this benchmark, and the right hand side of Figure 1 displays them graphically. Scores for the individual datasets constituting this benchmark can be found in Appendix E. We compare against the performance of MiniLM-L6-v2 and arctic-embed-xs. At the time of writing, these two models set the previous state-of-the-art for ≤30M parameter models[3]: MiniLM-L6-v2 achieves the best Borda score, while arctic-

[2] https://huggingface.co/spaces/mteb/leaderboard

[3] We have excluded from our comparisons models for which the training datasets are not known, although we outperform them too.

Table 1: nDCG@10 scores on the BEIR (Thakur et al., 2021) benchmark for `redact-ir`. †BM25 scores are obtained with $(k_1 = 0.9, b = 0.4)$. SPLADE++ and ColBERT v2 scores are from (Schlatt et al., 2025), while scores for arctic-embed-m-v1.5, arctic-embed-xs , and MiniLM-L6-v2 are from the public MTEB leaderboard.

	Size	ArguAna	ClimateFEVER	CQADupStack	DBPedia	FEVER	FiQA2018	HotpotQA	NFCorpus	NQ	Quora	SCIDOCS	SciFact	TREC-COVID	Touche2020	Avg.
`redact-ir (asym.)`	23M	**59.0**	**37.5**	**42.4**	**45.0**	**86.5**	**41.3**	68.5	**36.2**	**61.2**	86.0	20.3	70.2	**82.6**	30.1	**54.8**
`redact-ir`	23M	**58.4**	**34.6**	**42.3**	**44.6**	**86.6**	**38.4**	68.1	**35.8**	**58.9**	86.3	19.7	70.0	**80.3**	30.2	**53.9**
Comparisons																
arctic-embed-xs	23M	52.1	29.9	40.1	40.2	83.4	34.5	65.3	30.9	54.8	86.6	18.4	64.5	79.4	32.8	50.9
MiniLM-L6-v2	23M	50.2	20.3	41.3	32.3	51.9	36.9	46.5	31.6	43.9	**87.6**	**21.6**	64.5	47.2	16.9	42.3
BM25†	–	40.8	16.2	28.2	31.9	63.8	23.8	62.9	31.8	30.5	78.7	15.0	67.6	58.9	**44.2**	42.5
SPLADE++	110M	52.0	23.0	33.4	43.7	78.8	34.7	**68.7**	34.7	53.8	83.4	15.9	**70.4**	72.7	24.7	49.3
ColBERT v2	110M	45.3	17.6	35.9	44.1	77.4	34.6	66.5	33.0	54.7	85.1	15.0	69.1	73.2	25.7	48.4
Teacher																
arctic-embed-m-v1.5	109M	59.5	36.9	45.0	45.6	88.4	42.4	72.2	36.2	62.5	87.4	21.5	71.6	84.6	31.4	56.1

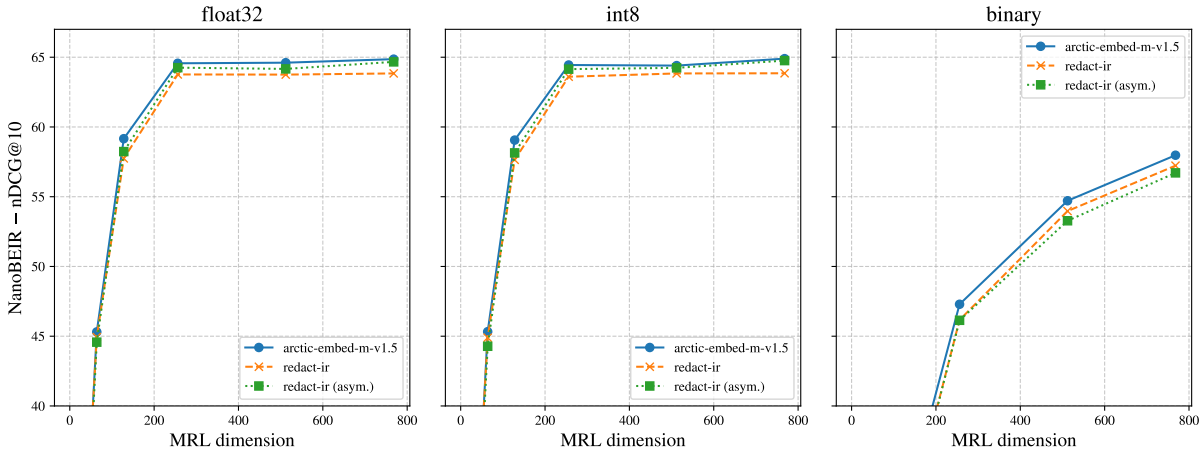


Figure 5: Performance curves for various MRL truncations, as well as output quantization regimes.

embed-xs has the highest average benchmark score at 56.1[4].

Our `redact-mt` model ranks no.1 on the public leaderboard of this benchmark for models of its size, at the time of writing. Specifically, our model sets a new state-of-the-art on 5 out of 7 tasks, as well as on the aggregated average benchmark score, achieving 59.4. Accordingly, our model retains 95.8% of its teacher performance despite the aggressive compression regime. Tests run on an AWS EC2 `i3.large` instance show that inference time speed-ups are of 24.4× / 23.7× for docs and queries respectively in this case. More details about these measurements are in Appendix G.

When run in asymmetric mode, the performance of our system on reranking and retrieval tasks is further improved: `redact-mt` sets a new sota on 6 out 7 tasks, and its aggregated benchmark score is increased to 60.1 (96.9% of its teacher).

This model also inherits MRL and vector quantization robustness from its teacher. The performance profiles are similar to those shown in Figure 5, see Appendix H.

4.3 Robustness Margins

Throughout this work we consider knowledge distillation as an approximation scheme, where a smaller model attempts to approximate the embedding map of its teacher. We consequently introduce the ℓ_2 approximation error in Eq. (3) as the loss to drive our training scheme.

In this subsection we investigate how much approximation error can a retrieval system based on `redact` models sustain without excessive perfor-

[4]MTEB v2 scores are an aggregate, dimensionless score.

Table 2: Scores on the MTEB v2 (English) benchmark for `redact-mt`. † these scores are identical between standard and asymmetric mode.

# Datasets	Size	Class. 8	Cluster. 8	Pair Class. 3	Rerank 2	Retrieval 10	STS 9	Summ. 1	Avg.	Avg. (Datasets)
`redact-mt` (asym.)	23M	**76.9**	**46.5**	**84.5**	**47.3**	51.8	**82.8**	**30.9**	**60.1**	**64.1**
`redact-mt`	23M	†	†	†	46.9	47.3	†	†	**59.4**	**63.0**
Comparisons										
MiniLM-L6-v2	23M	69.3	44.9	82.4	**47.1**	42.9	79.0	26.0	55.9	59.0
arctic-embed-xs	23M	67.0	42.4	81.3	45.3	**52.7**	76.2	28.0	56.1	59.8
Teacher										
mxbai-l-v1	335M	79.1	47.5	87.2	48.1	55.4	84.4	32.6	62.0	66.3

mance degradation.

To this end, Figure 6 shows the downstream performance of several `redact-ir` checkpoints we stored during training at the end of each epoch[5]. In these results, `redact-ir` was used to encode both queries as well as documents. As can be seen, the lowest average approximation error on the validation set $|I^{\text{val}}|^{-1} \sum_{i \in I^{\text{val}}} \epsilon_i$ we were able to obtain is ≈ 0.3, where $\epsilon_i = \|y_i - \hat{y}_i\|_2$. Consider that `redact-ir`'s and its teacher embeddings have an ℓ_2-norm of 1, meaning that $0 \leq \epsilon_i \leq 2$. As such, the residual approximation error is significant even with our best checkpoints.

On the other hand, Figure 6 also plots a linear trend for our checkpoints. Besides confirming the intuitive idea that lower approximation errors lead to scores that are closer to the teacher's reference performance, this curve suggests that it is not necessary for this approximation error to be 0 for there to be no distinguishable performance difference with respect to the teacher. We graphically represent the region where we hypothesize this to occur as the shaded region in Figure 6 and refer to it as the *robustness margin* of the system. These results are analogous for `redact-mt`, see Appendix F.

We believe that the success of our scheme is owed to these robustness margins.

5 Conclusion

This work introduced a lightweight knowledge distillation regime that produces text embedding models *aligned* to their teachers. These models set a new state-of-the-art on information retrieval and multitask benchmarks respectively, ranking no.1 on the corresponding leaderboards for models of their size. Further, to the best of our knowledge ours are the first publicly available models that enable the flexible asymmetric architecture shown in Figure 2, and that this architecture can enable further retrieval performance gains. We are also the first to show that this distillation procedure automatically inherits MRL and vector quantization properties from the teacher.

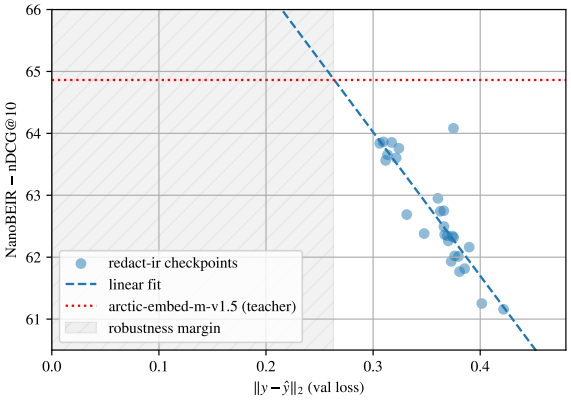


Figure 6: NanoBEIR performance of various `redact-ir` training checkpoints.

Limitations

The models we published have only been trained and evaluated on the English language. Whether such compact models can accommodate multilinguality (including programming languages) is an open question.

We also have not investigated the scaling of our procedure, i.e., whether larger models can be synthesized as effectively. Many recent embedding models use decoders instead of encoders, and whether the technique needs to be adapted in this case also needs further investigation.

Finally, we have adapted and incorporated several other distillation losses in our experiments,

[5]We removed the checkpoints from the first epoch of each cycle as we found downstream performance at high learning rates to display a high degree of variability.

such as MiniLM, TinyBERT and DistilBERT, as reported in Appendix B, but weren't able to materialize concrete improvements in effectiveness. This despite the fact that these additions take advantage of more information from the teacher, in the form of its internal representations and attention matrices. More experimentation may shed further light on this counterintuitive result.

References

Daniel Campos, Alessandro Magnani, and ChengXiang Zhai. 2023. Quick dense retrievers consume kale: Post training kullback leibler alignment of embeddings for asymmetrical dual encoders. *arXiv preprint arXiv:2304.01016*.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. 2020. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org.

Xuanang Chen, Ben He, Kai Hui, Le Sun, and Yingfei Sun. 2021. Simplified tinybert: Knowledge distillation for document retrieval. In *Advances in Information Retrieval: 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 – April 1, 2021, Proceedings, Part II*, page 241–248, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

Kenneth Enevoldsen, Isaac Chung, Imene Kerboua, Márton Kardos, Ashwin Mathur, David Stap, Jay Gala, Wissam Siblini, Dominik Krzemiński, Genta Indra Winata, Saba Sturua, Saiteja Utpala, Mathieu Ciancone, Marion Schaeffer, Gabriel Sequeira, Diganta Misra, Shreeya Dhakal, Jonathan Rystrøm, Roman Solomatin, and 67 others. 2025. Mmteb: Massive multilingual text embedding benchmark. *arXiv preprint arXiv:2502.13595*.

Hugging Face. 2023. Distilbert. Accessed: 2025-05-22.

Thibault Formal, Benjamin Piwowarski, and Stéphane Clinchant. 2021. *SPLADE: Sparse Lexical and Expansion Model for First Stage Ranking*, page 2288–2292. Association for Computing Machinery, New York, NY, USA.

Mansi Gupta, Nitish Kulkarni, Raghuveer Chanda, Anirudha Rayasam, and Zachary C Lipton. 2019. Amazonqa: A review-based question answering task. *arXiv preprint arXiv:1908.04364*.

Sebastian Hofstätter, Sophia Althammer, Michael Schröder, Mete Sertkan, and Allan Hanbury. 2020. Improving efficient neural ranking models with cross-architecture knowledge distillation. *Preprint*, arXiv:2010.02666.

Hugging Face. 2024. Model distillation example for sentence transformers. https://github.com/huggingface/sentence-transformers/blob/main/examples/sentence_transformer/training/distillation/model_distillation.py. Accessed: 2025-12-18.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2020. TinyBERT: Distilling BERT for natural language understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4163–4174, Online. Association for Computational Linguistics.

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2567–2577.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. triviaqa: A Large Scale Distantly Supervised Challenge Dataset for Reading Comprehension. *arXiv e-prints*, arXiv:1705.03551.

Kaggle. 2023. Dataset containing 479k english words.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2022. Matryoshka representation learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 30233–30249. Curran Associates, Inc.

Luke Merrick, Danmei Xu, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed: Scalable, efficient, and accurate text embedding models. *arXiv preprint arXiv:2405.05374*.

Niklas Muennighoff, Hongjin SU, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2025. Generative representational instruction tuning. In *The Thirteenth International Conference on Learning Representations*.

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, Johannes Heidecke, Pranav Shyam, Boris Power, Tyna Eloundou Nekoul, Girish Sastry, Gretchen Krueger, David Schnurr, Felipe Petroski Such, Kenny Hsu, and 6 others. 2022. Text and code embeddings by contrastive pre-training. *Preprint*, arXiv:2201.10005.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2016. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

Jakob Nielsen. 1994. *Usability engineering*. Morgan Kaufmann.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4512–4525, Online. Association for Computational Linguistics.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2020. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *Preprint*, arXiv:1910.01108.

Keshav Santhanam, Omar Khattab, Jon Saad-Falcon, Christopher Potts, and Matei Zaharia. 2022. Colbertv2: Effective and efficient retrieval via lightweight late interaction. *Preprint*, arXiv:2112.01488.

Ferdinand Schlatt, Tim Hagen, Martin Potthast, and Matthias Hagen. 2025. Tite: Token-independent text encoder for information retrieval. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '25, page 2493–2503, New York, NY, USA. Association for Computing Machinery.

Hongjin Su, Weijia Shi, Jungo Kasai, Yizhong Wang, Yushi Hu, Mari Ostendorf, Wen-tau Yih, Noah A. Smith, Luke Zettlemoyer, and Tao Yu. 2023. One embedder, any task: Instruction-finetuned text embeddings. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1102–1121, Toronto, Canada. Association for Computational Linguistics.

Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. 2020. Mobilebert: a compact task-agnostic bert for resource-limited devices. *arXiv preprint arXiv:2004.02984*.

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

Minh-Phuc Truong, Hai An Vu, Tu Vu, Nguyen Thi Ngoc Diep, Linh Ngo Van, Thien Huu Nguyen, and Trung Le. 2025. EMO: Embedding model distillation via intra-model relation and optimal transport alignments. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 7605–7617, Suzhou, China. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. 2020. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788.

Xhluca. 2023. Bm25 benchmarks. https://github.com/xhluca/bm25-benchmarks/blob/22df0ccccc083d2985a5b7e55d3ecd1764d4f9ed/README.md?plain=1#L220. Accessed: 2025-05-07.

Jinsung Yoon and Sercan O Arik. 2025. Embedding-converter: A unified framework for cross-model embedding transformation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 25464–25482, Vienna, Austria. Association for Computational Linguistics.

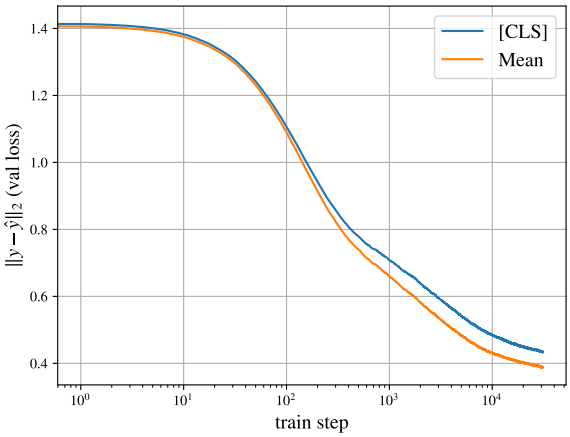
Puxuan Yu, Luke Merrick, Gaurav Nuti, and Daniel Campos. 2024. Arctic-embed 2.0: Multilingual retrieval without compromise. *Preprint*, arXiv:2412.04506.

A Training Setup

A.1 Pooling Layer

In Figure 7 a comparison of different popular pooling layers is shown. These are obtained by running one training epoch of `redact-ir` on the entire training dataset. We observe that mean pooling produces substantially lower (better) validation scores than `[CLS]` at the end of the training epoch.

Figure 7: Pooling layer comparison. Mean pooling achieves the lowest (best) final validation loss.



A.2 Learn Rate Decay Schedule

Figure 8 shows a comparison of different popular learning rate decay schedules. Each point reflects a run of 10 training epochs of `redact-ir`, where each epoch contains a number of training batches reported on the x-axis. On the y-axis we have the final validation loss. For the constant variation, we keep lr=$1e-4$ throughout the 10 epochs. For linear decay, we decrease the learning rate at the end of each epoch, from lr=$1e-4$ to lr=$1e-5$ at the 10th epoch. Cosine annealing steps the learning rate down each training step.

Constant learning rate achieves the lowest (best) validation scores at low training data regimes, while linear decay has the best loss amongst the rate schedules we compared in the presence of more training data, when each epoch consist of 100k training batches, more closely matching our production runs.

A.3 Batch Size Selection

Table 3 reports our results when running one training epoch of `redact-ir` under different batch sizes. The table contains both batch size as well as total number of batches: the amount of training data is kept constant so smaller batch sizes lead to a larger number of training batches. The wall clock time to complete the epoch is also reported; training is conducted on a single A100 40GB GPU. The validation loss reported is the loss we measure at the end of the training epoch.

Figure 8: Comparison of learning rate decay schedules. Linear decay ends with the lowest (best) final validation loss when the training data is most abundant in this comparison, more closely resembling our production runs.

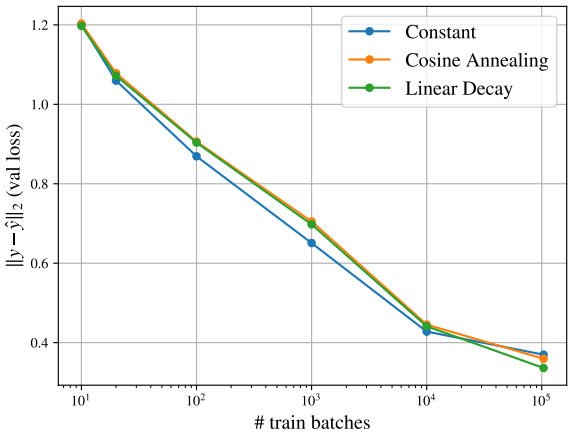


Table 3: Training metrics for different batch sizes. Batch size of 32 has the best trade-off between epoch completion time and dev loss at the end of the epoch.

Batch Size	Time	Val Loss ℓ_2	# Batches
16	4h 19min	0.4194	400k
32	3h 05min	0.4214	200k
64	2h 51min	0.4301	100k
128	2h 40min	0.4390	50k
256	2h 32min	0.4593	25k

One can observe that the final losses for batch sizes 16 and 32 are similar, but as we further increase the batch sizes the final loss degrades more and more, while the training wall clock time does not improve as much. We find the sweetspot between wall clock time and final validation loss at a batch size of 32.

This result shows that our scheme generally favors more steps with smaller batch sizes, rather than the other way around. This makes intuitive sense as the model needs to be completely reoriented to match the embedding vector space of the target teacher model.

B Alternative Knowledge Distillation Approaches

We implemented the losses of three popular knowledge distillation frameworks: MiniLM (Wang et al., 2020), TinyBERT (Jiao et al., 2020) and DistilBERT (Sanh et al., 2020).

All of these frameworks are designed for the distillation of language models, i.e., transformer architectures equipped with a language modelling head. Text embedding models based on transformers follow the standard architecture discussed in Section 3 and usually remove the language modelling head and replace it with a pooling layer and an optional normalization layer. We additionally add a linear layer $W^{\text{out}} \in \mathbb{R}^{d' \times d}$, to map `redact`'s embedding dimension d' to the desired teacher output dimension d. As shown below, none of the losses proposed by these knowledge distillation procedures would entail a training signal for W^{out}. To address this, we combined each loss with our ℓ_2 distillation loss in Eq. (3). The inclusion of this loss component is also necessary to ensure that the models distilled are *aligned* to their teacher, a core goal of this work.

Further, as proposed in their original papers, these knowledge distillation schemes require the tokenizers of the teacher and the student to match. MiniLM and TinyBERT further require the number of attention heads to match. Our scheme does not have these restrictions.

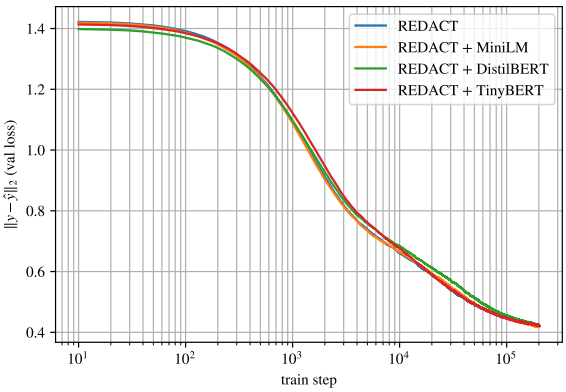
We run one epoch with each one of these distillation schemes when combined with our output alignment loss Eq. (3). Figure 9a shows the validation loss curves over this epoch, while Figure 9b zooms in on the last 100k steps. The final approximation errors $\|y - \hat{y}\|_2$ on the validation set we obtain with all these schemes is very similar, as shown in the Figures.

The downstream performance on NanoBEIR reported in Table 4, however, suggests that none of these additions produces better results than working with just the loss in Eq. (3).

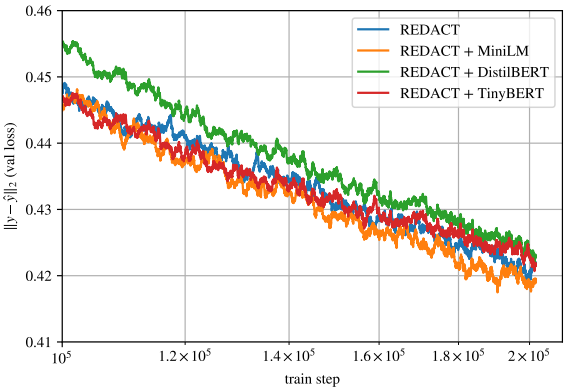
Below we provide more details about how we adapted these methods in our implementation.

B.1 MiniLM

We note that for this training procedure to work as proposed in (Wang et al., 2020), the number of attention heads in the student and the teacher models have to match. This is not required by our loss (3).



(a) Validation loss over the entire 1st epoch



(b) Zoom in on the last 100k steps

Figure 9: Comparison of the validation losses observed over the first training epoch under different knowledge distillation approaches discussed in Appendix B. The difference in validation losses measured at the end of the first training epoch is insubstantial.

Table 4: Comparison of different knowledge distillation losses measured on the NanoMSMARCO dataset, after one training epoch. These results indicate that additional training signals leveraging Transformer's internal representations may not lead to better downstream results.

loss	nDCG@10
$\mathcal{L}_{\text{REDACT}}$	60.7
$\mathcal{L}_{\text{REDACT}} + \mathcal{L}_{\text{MiniLM}}$	54.9
$\mathcal{L}_{\text{REDACT}} + \mathcal{L}_{\text{TinyBERT}}$	53.7
$\mathcal{L}_{\text{REDACT}} + \mathcal{L}_{\text{DistilBERT}}$	55.3

Given the value vector $\mathrm{V}_{l,a} \in \mathbb{R}^{T\times C}$ for layer l and attention head a, where $T \in \mathbb{N}$ is the sequence length and $C \in \mathbb{N}$ is the number of channels in the value vector of attention a; typically, $C = d/A$, where $d \in \mathbb{N}$ is the hidden dimension of the Transformer and $A \in \mathbb{N}$ is the number of heads. We disregard the batch dimension in this analysis. The *value relation matrix* $\mathrm{VR}_{l,a} \in \mathbb{R}^{T\times T}$ is then defined as

$$\mathrm{VR}_{l,a} = \mathrm{softmax}\left(\frac{\mathrm{V}_{l,a}\cdot \mathrm{V}_{l,a}^{\intercal}}{\sqrt{C}}\right). \tag{4}$$

Let L and L' be the number of transformer layers in the teacher and student models respectively. Then,

$$\mathcal{L}_{\mathrm{VR}} = \frac{1}{AT}\sum_{t=1}^{T}\sum_{a=1}^{A}\mathcal{D}_{\mathrm{KL}}(\mathrm{VR}^{\tau}_{L,a}\,||\,\mathrm{VR}^{\sigma}_{L',a})_t, \tag{5}$$

where $\mathcal{D}_{\mathrm{KL}}$ is the KL-divergence between the value relation (4) of the teacher τ and student σ. According to Eq. (5), loss is computed only for the last transformer layers of the respective networks L and L', and averaged over the attention heads and input tokens.

An additional loss term is added to align the attention patterns of the two transformer backbones. Namely, let

$$\mathrm{Att}_{l,a} = \mathrm{softmax}\left(\frac{\mathrm{K}_{l,a}\cdot \mathrm{Q}_{l,a}^{\intercal}}{\sqrt{C}}\right) \tag{6}$$

where $\mathrm{K}_{l,a} \in \mathbb{R}^{T\times C}$ are the keys and $\mathrm{Q}_{l,a} \in \mathbb{R}^{T\times C}$ are The queries in head a of layer l of the transformer. $\mathrm{Att}_{l,a} \in \mathbb{R}^{T\times T}$ is thus the attention matrix in the given transformer head. Then, similarly to Eq.(5), we define

$$\mathcal{L}_{\mathrm{Att}} = \frac{1}{AT}\sum_{t=1}^{T}\sum_{a=1}^{A}\mathcal{D}_{\mathrm{KL}}(\mathrm{Att}^{\tau}_{L,a}\,||\,\mathrm{Att}^{\sigma}_{L',a})_t, \tag{7}$$

as the KL-divergence between the student's σ and the teacher τ attentions in the last transformer layer, averaged over the attention heads and input tokens.

The loss we adopt to produce the results in Table 4 is then

$$\mathcal{L}_{\mathrm{MiniLM}} = \mathcal{L}_{\mathrm{VR}} + \mathcal{L}_{\mathrm{Att}}. \tag{8}$$

B.2 TinyBERT

TinyBERT's (Jiao et al., 2020) proposed knowledge distillation scheme entails a soft cross-entropy loss between teacher and student output logits. However, as remarked earlier, we discard the language modelling head in our architecture, and hence discard this loss component too.

We next discuss our implementation of the other loss components proposed in TinyBERT.

Let $h_{l,t} \in \mathbb{R}^d$ be the hidden state at the l-th layer and at the t-th token. As a convention, we denote $h_{0,t}$ as the hidden state before the first transformer layer; in BERT, $h_{0,t}$ is the superposition of the input and positional embeddings.

To align hidden states, TinyBERT proposes the following loss:

$$\mathcal{L}_{\mathrm{Hidn}} = \frac{1}{TL}\sum_{t=1}^{T}\sum_{l=1}^{L'}\mathrm{MSE}(h^{\tau}_{g(l),t} - W^{\mathrm{map}}h^{\sigma}_{l,t}), \tag{9}$$

where $g : \mathbb{N} \to \mathbb{N}$ maps a hidden layer index in the student model to a corresponding hidden layer in the teacher and is such that $g(0) = 0$ and $g(L') = L$. As proposed in (Wang et al., 2020), we use $g(l) = \lfloor \frac{L}{L'} \rfloor l$. $W^{\mathrm{map}} \in \mathbb{R}^{d'\times d}$ maps the student's internal representations, which are often of smaller dimensions, into the teacher ones.

Similar to Section B.1, TinyBERT also adopts a loss to align the attention patterns as follows:

$$\mathcal{L}_{\mathrm{Att}} = \frac{1}{AL}\sum_{a=1}^{A}\sum_{l=1}^{L'}\mathrm{MSE}(\widetilde{\mathrm{Att}}^{\tau}_{g(l),a}, \widetilde{\mathrm{Att}}^{\sigma}_{l,a}), \tag{10}$$

where $\widetilde{\mathrm{Att}}$ is the attention matrix defined in Eq. (6) but without $\mathrm{softmax}(\cdot)$. Because of this loss component, TinyBERT (Jiao et al., 2020) requires the number of heads in the student's and teacher's transformer layers to match, like with the MiniLM loss discussed in Section B.1.

The aggregated loss we use to implement this knowledge distillation strategy is then

$$\mathcal{L}_{\mathrm{TinyBERT}} = \mathcal{L}_{\mathrm{Hidn}} + \mathcal{L}_{\mathrm{Att}}. \tag{11}$$

B.3 DistilBERT

DistilBERT (Sanh et al., 2020) employes a combined loss on output logits and cosine similarity on internal states. As before, we discard the loss on the logits because we do not have a language modelling head. The paper does not specify how to compute the loss component related to the hidden states when the number of layers do not match. The implementation at (Face, 2023) suggests that this loss is only applied to the last layer. It also requires hidden state dimensions to match. We compensate for this latter limitation by applying a linear map like in Eq. (7).

We thus have

$$\mathcal{L}_{\text{DistilBERT}} = -\cos\_\text{sim}(h^{\tau}_{L,t}, W^{\text{map}} h^{\sigma}_{L',t}) \tag{12}$$

where $\cos\_\text{sim}(\cdot)$ is the vector cosine similarity.

C The Vocabulary Dataset

We took the list of 479k words from (Kaggle, 2023) and used Claude 3.5 Sonnet to produce definitions and important facts about them using the prompt in Listing 1. Listing 2 shows some examples from this dataset.

D Scoring Examples

Table 5 shows a comparison of top 10 documents as retrieved by the reference teacher model arctic-embed-m-v1.5, and compared to the results from `redact-ir` run in asymmetric and standard modes.

The documents are retrieved from 100k documents from the CC-News dataset samples of 2024. The query is "best marvel movie". As can be seen, when run in asymmetric mode there is an 8/10 overlap with the teacher's retrieved documents, while in standard mode the overlap is 9/10. The top retrieval result, which has a score substantially higher than the rest, is the same for all three methods.

E MTEB v2 (Eng) Results

Table 6 reports the results of the MTEB v2 (English) benchmark on the 41 individual datasets that constitute it.

Results for `redact-mt` standard and asymmetric modes are identical except on reranking and retrieval tasks, as these are the only tasks that have query and document sides.

Highlighted in bold are cases where the the distilled model performs better than the teacher.

Listing 1: Prompt used to produce the vocab dataset.

```
Please provide ALL the definitions and, if
    available, important facts about the
    following terms:

%s

Provide these definitions/facts as a JSON
    document, formatted as:
{
 "word": ["definition or fact 1", "definition or
      fact 2", "definition or fact 3", "
     definition or fact 4", ...]
}

If the word has multiple definitions or facts
    attached to it, there should be a string for
     each definition or fact. The list of
    definitions should be complete.

The definition or fact should contain the term
    in question, so it can be read stand-alone.
    For example, the output expected for the
    terms "subfigure", "linking" and "Pinckney"
    is:

{
 "subfigure": [
   "A subfigure is a secondary or smaller figure
        within a larger figure or diagram,
       often used in academic and scientific
       publications."
 ],
 "linking": [
   "Linking is the process of connecting or
       joining two or more things together.",
   "In computing, linking is the process of
       combining various pieces of code and
       data into a single executable program."
 ],
 "Pinckney": [
   "Pinckney refers to Charles Cotesworth
       Pinckney, an American statesman and
       diplomat.",
   "Pinckney refers to place names or other
       historical figures with the surname
       Pinckney.",
   "Pinckney's Treaty, also known as the Treaty
       of San Lorenzo or the Treaty of Madrid,
       was a diplomatic agreement signed on
       October 27, 1795, between the United
       States and Spain. The treaty defined the
        border between the United States and
       Spanish Florida at 31 N latitude."
 ]
}

There should not be any reference between any
    definition of or facts about a term, so they
     should NEVER contain the words such as 'can
     also refer to' or 'also refers to'.

Provide your output as a JSON object, and
    nothing else than the JSON. Do not include
    any additional descriptions or introductory
    text.
```

Table 5: Top 10 documents and their scores as retrieved from a collection of 100k news articles by the teacher model, and compared with `redact-ir` run in asymetric and standard modes. A substantial overlap in the documents retrieved and score similarities can be observed.

Score	Document
	teacher (ground truth)
0.5904	The MCU Movie With The Highest Rotten Tomatoes Score - Looper: The MCU Movie With The Highest Rotten...
0.5187	Kevin Feige: Marvel maestro, box office bellwether: Is Kevin Feige good for the film industry? Depen...
0.4752	Netflix top 10 movies — here's the 3 worth watching right now: The Netflix top 10 is a great resourc...
0.4721	X-Men '97 Created A New Challenge For Marvel's Future: X-Men '97 has taken the world by storm. The D...
0.4642	The Best Movies of 2024 So Far: Summer is here! In the old days, that would mean heading to the mult...
0.4632	'Indrani' is like a massified version of Marvel movies: Makers: 'Indrani' is like a massified versio...
0.4594	5 best movies to watch this weekend on Netflix, Prime Video, Hulu and more: As we head toward summer...
0.4535	Marvel's first immersive story for the Apple Vision Pro is the most fun I've had on the device: Yes,...
0.4453	Netflix has just added one of 2023's very best movies: After its cinema release just last December, ...
0.4420	Former MCU star just well and truly destroyed rumors of their 'Avengers' comeback, and we're so glad...
	`redact-ir` (asym.)
0.5905	The MCU Movie With The Highest Rotten Tomatoes Score - Looper: The MCU Movie With The Highest Rotten...
0.5079	Kevin Feige: Marvel maestro, box office bellwether: Is Kevin Feige good for the film industry? Depen...
0.4729	X-Men '97 Created A New Challenge For Marvel's Future: X-Men '97 has taken the world by storm. The D...
0.4621	Netflix top 10 movies — here's the 3 worth watching right now: The Netflix top 10 is a great resourc...
0.4618	Marvel's first immersive story for the Apple Vision Pro is the most fun I've had on the device: Yes,...
0.4548	'Indrani' is like a massified version of Marvel movies: Makers: 'Indrani' is like a massified versio...
0.4491	5 best movies to watch this weekend on Netflix, Prime Video, Hulu and more: As we head toward summer...
0.4466	The Best Movies of 2024 So Far: Summer is here! In the old days, that would mean heading to the mult...
0.4394	Netflix has just added one of 2023's very best movies: After its cinema release just last December, ...
0.4356	One of the best hidden gem movies of last year is now streaming on Netflix: Godzilla Minus One tease...
	`redact-ir`
0.5925	The MCU Movie With The Highest Rotten Tomatoes Score - Looper: The MCU Movie With The Highest Rotten...
0.5312	X-Men '97 Created A New Challenge For Marvel's Future: X-Men '97 has taken the world by storm. The D...
0.5238	Kevin Feige: Marvel maestro, box office bellwether: Is Kevin Feige good for the film industry? Depen...
0.4908	'Indrani' is like a massified version of Marvel movies: Makers: 'Indrani' is like a massified versio...
0.4823	Netflix top 10 movies — here's the 3 worth watching right now: The Netflix top 10 is a great resourc...
0.4681	One of the best hidden gem movies of last year is now streaming on Netflix: Godzilla Minus One tease...
0.4648	Marvel's first immersive story for the Apple Vision Pro is the most fun I've had on the device: Yes,...
0.4608	5 best movies to watch this weekend on Netflix, Prime Video, Hulu and more: As we head toward summer...
0.4554	Netflix has just added one of 2023's very best movies: After its cinema release just last December, ...
0.4505	The Best Movies of 2024 So Far: Summer is here! In the old days, that would mean heading to the mult...

Listing 2: Examples from the vocab dataset.

```
{
  "1080": [
    "1080 is a natural number following 1079 and
        preceding 1081.",
    "1080 refers to a video display resolution of
         1920x1080 pixels, also known as Full HD
        .",
    "1080 is the chemical compound sodium
        fluoroacetate, used as a pesticide.",
    "In mathematics, 1080 is the sum of four
        consecutive primes (263 + 269 + 271 +
        277)."
  ],
  "Abipon": [
    "Abipon refers to an indigenous people who
        historically inhabited the Gran Chaco
        region of Argentina.",
    "The Abipon were known for their horsemanship
         and warrior culture.",
    "Abipon is an extinct language once spoken by
         the Abipon people."
  ],
  "abaxial": [
    "Abaxial in botany refers to the surface of a
         leaf or other organ facing away from
        the axis or stem.",
    "Abaxial is the opposite of adaxial in plant
        anatomy."
  ]
}
```

In many of these cases, these differences are negligible except for Touche2020Retrieval.v3 and StackExchangeClustering.v2. We have manually checked the former and verified that the results are correct.

F Robustness Margins for `redact-mt`

Section 4.3 discusses our observed systems' robustness to the approximation error intorduced by our knowledge distillation scheme. The robustness margin for `redact-mt` is shown in Figure 10. Note that the teacher model for `redact-mt`, mxbai-l-v1, does not output normalized vectors, and as such the approximation error ϵ does not have an upper bound of 2.

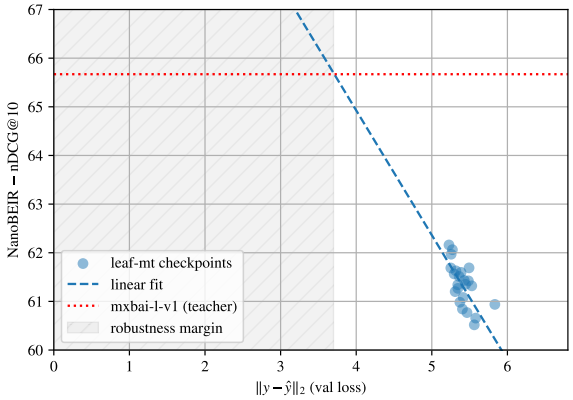
G Inference Time Performance (CPU)

Increasing inference throughput, lowering latencies, and operating on more economical infrastructure are typical core goals of model distillation activities.

Table 7 reports our results for these key metrics of `redact-ir` and `redact-mt` when compared to their corresponding teachers.

The experiments are conducted on an AWS EC2 `i3.large` instance, which is a commonly used CPU-only VM for database and search workloads. These instances are equipped with 2 vCPUs and 15.25GB of RAM. Inference is performed on the ONNX Runtime.

Figure 10: NanoBEIR performance of various checkpoints stored while training `redact-mt`. Similar to the discussion in Section 4.3, we observe substantial robustness margins for this model.



To obtain these measurements, we first select the following set of batch sizes: 1, 2, 4, 8, 16 and 24. For each batch size, we randomly sample a corresponding number of queries and documents from MSMARCO. Queries are generally much shorter than documents in this dataset; they reflect the typical user queries of an online search engine. We use Python's `timeit` package to measure the wall clock time required to complete the inference of a batch 7 times. As shown in Figure 11, inference time scales approximately linearly with batch size. We then divide this time by the batch size to derive the throughput metrics (docs/sec and queries/sec) reported in the Table. Mean and standard deviation for these throughput figures are computed across all the samples for all the batch sizes.

Our results show that for `redact-ir`, the 4.7× parameter reduction leads to a 6.5× throughput increase when used to encode documents, and a 7.3× throughput increase for queries. For `redact-mt`, on the other hand, the 14.6× parameter reduction leads to a 24.4× throughput increase for docs, and 23.7× throughput increase for queries.

Table 7 also reports the minimum latency, i.e., the shortest amount of time a user would need to wait for inference to complete. In all of our experiments, this occurs at a batch size of 1. Industry research and usability studies, including (Nielsen, 1994), commonly highlight 0.1 seconds as the re-

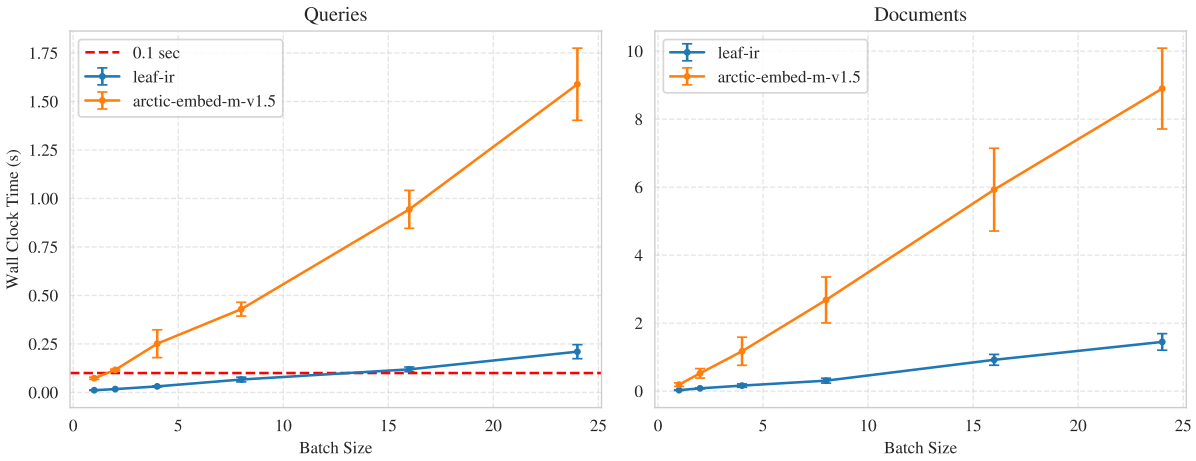
Table 6: MTEB v2 (English) benchmark performance. `redact-mt` substantially outperforms the teacher model on Touche2020Retrieval.v3 and StackExchangeClustering.v2. Highlighted are datasets in which `redact-mt` outperforms the teacher. (*) These values are identical in standard and asymmetric modes. †mxbai-l-v1 values are taken from the online version of the MTEB leaderboard.

Task Name	redact-mt	redact-mt (asym.)	mxbai-l-v1†	Metric	Task Type
AmazonCounterfactualClassification	71.27	*	75.07	Accuracy	Classification
Banking77Classification	85.07	*	87.80	Accuracy	Classification
ImdbClassification	89.05	*	92.83	Accuracy	Classification
MTOPDomainClassification	92.73	*	93.95	Accuracy	Classification
MassiveIntentClassification	72.35	*	76.22	Accuracy	Classification
MassiveScenarioClassification	77.04	*	79.89	Accuracy	Classification
ToxicConversationsClassification	67.50	*	67.31	Accuracy	Classification
TweetSentimentExtractionClassification	**60.23**	*	**59.70**	**Accuracy**	**Classification**
ArXivHierarchicalClusteringP2P	59.50	*	60.11	V Measure	Clustering
ArXivHierarchicalClusteringS2S	52.91	*	58.99	V Measure	Clustering
BiorxivClusteringP2P.v2	41.76	*	41.87	V Measure	Clustering
MedrxivClusteringP2P.v2	**37.58**	*	**37.27**	**V Measure**	**Clustering**
MedrxivClusteringS2S.v2	34.49	*	34.97	V Measure	Clustering
StackExchangeClustering.v2	**58.34**	*	**55.08**	**V Measure**	**Clustering**
StackExchangeClusteringP2P.v2	40.28	*	40.47	V Measure	Clustering
TwentyNewsgroupsClustering.v2	47.29	*	51.10	V Measure	Clustering
SprintDuplicateQuestions	96.48	*	96.82	AP	PairClassification
TwitterSemEval2015	71.25	*	78.55	AP	PairClassification
TwitterURLCorpus	85.64	*	86.23	AP	PairClassification
AskUbuntuDupQuestions	61.35	62.03	63.49	MAP	Reranking
MindSmallReranking	32.35	32.53	32.62	MAP	Reranking
ArguAna	61.64	64.44	65.47	nDCG@10	Retrieval
CQADupstackGamingRetrieval	54.29	54.00	58.94	nDCG@10	Retrieval
CQADupstackUnixRetrieval	36.58	37.25	41.77	nDCG@10	Retrieval
ClimateFEVERHardNegatives	26.79	35.96	36.23	nDCG@10	Retrieval
FEVERHardNegatives	72.49	81.13	86.54	nDCG@10	Retrieval
FiQA2018	36.27	42.22	45.27	nDCG@10	Retrieval
HotpotQAHardNegatives	62.23	65.07	72.50	nDCG@10	Retrieval
SCIDOCS	18.09	19.06	23.10	nDCG@10	Retrieval
TRECCOVID	52.52	73.03	75.53	nDCG@10	Retrieval
Touche2020Retrieval.v3	**51.58**	**45.72**	**48.60**	**nDCG@10**	**Retrieval**
BIOSSES	84.45	*	86.05	Spearman	STS
SICK-R	80.87	*	82.78	Spearman	STS
STS12	77.96	*	79.07	Spearman	STS
STS13	87.93	*	89.79	Spearman	STS
STS14	82.40	*	85.22	Spearman	STS
STS15	88.08	*	89.34	Spearman	STS
STS17	87.12	*	89.21	Spearman	STS
STS22.v2	**69.12**	*	**69.04**	**Spearman**	**STS**
STSBenchmark	87.19	*	89.29	Spearman	STS
SummEvalSummarization.v2	30.86	*	32.63	Spearman	Summarization

Table 7: Performance comparison across different models. Max N is the largest batch size that can be computed in 100ms or less; "-" indicates that no batch size can be computed within this time constraint.

	docs				queries			
model	**docs/s**	**speedup**	**min latency**	**max N**	**queries/s**	**speedup**	**min latency**	**max N**
arctic-embed-m-v1.5	3.2 ± 0.8	$1\times$	191 ± 52.6 ms	-	16.7 ± 1.4	$1\times$	73.2 ± 5.8 ms	1
`redact-ir`	21.9 ± 5.5	$6.5\times$	28.2 ± 9.69 ms	2	121.3 ± 14.0	$7.3\times$	11.4 ± 1.38 ms	8
mxbai-l-v1	0.9 ± 0.2	$1\times$	834 ± 298 ms	-	4.9 ± 0.4	$1\times$	236 ± 18.5 ms	-
`redact-mt`	21.4 ± 5.8	$24.4\times$	39.2 ± 10.2 ms	2	117.0 ± 15.9	$23.7\times$	11.5 ± 1.2 ms	8

Figure 11: `redact-ir` wall clock inference times on an `i3.large` instance for queries and documents, as a function of batch size. `redact` models allow for larger batch sizes while remaining under the 0.1 seconds threshhold for queries.



sponsiveness threshold for cognitive perception in user-facing applications. In the Table we thus also report the largest batch size that can be processed while remaining below this threshold. We report these metrics both for queries as well as documents, although this measurement is more critical for queries. We also show this as a red dashed line in Figure 11. A dash "-" in the table indicates that no batch size exists that can be computed under 0.1 seconds.

As shown, `redact` models have substantially lower minimum latencies than their counterpart teachers. They are also the only models that can process a batch size within the 0.1 seconds threshold on our test hardware configuration, with the exception of arctic-embed-m-v1.5 when processing queries.

H MRL Results for `redact-mt`

Figure 12 shows that this model also acquires MRL and quantization properties of its teacher. The MRL curves appear to have a more gradual performance profile than those shown in Figure 5, owing to differences in training for these properties in the corresponding teacher models. This figure illustrates that our approach is able to inherit MRL and vector quantization independently of how these properties were trained for originally by the teacher.

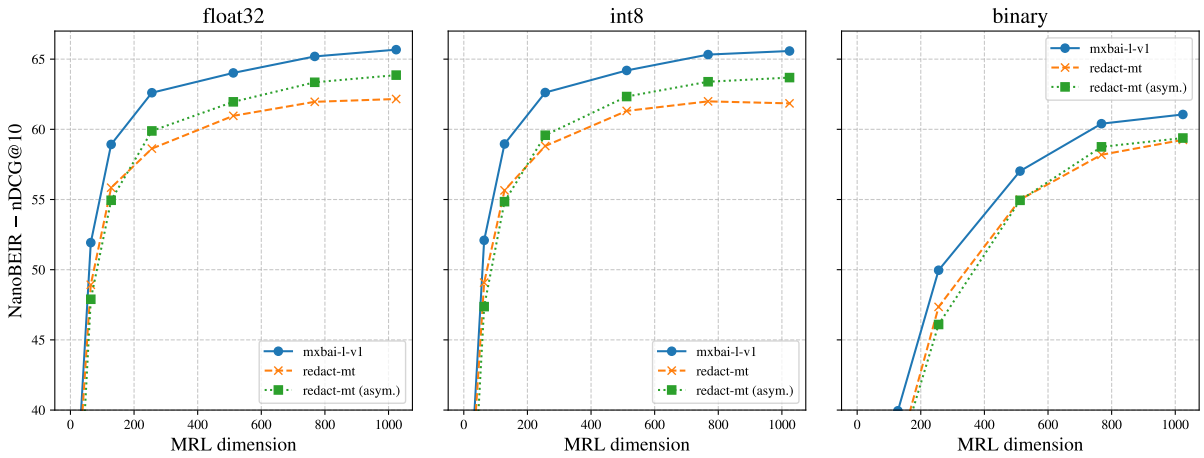


Figure 12: Performance curves across various embedding dimensions with a teacher model trained with MRL (Kusupati et al., 2022), as well as output quantization regimes. Benchmarked according to NanoBEIR. The Figure shows that our model `redact-mt` inherits MRL and vector quantization properties from its teacher.