DREAM: Dense Retrieval Embeddings via Autoregressive Modeling

Yixuan Tang, Yi Yang
The Hong Kong University of Science and Technology
ytangch@connect.ust.hk, imyiyang@ust.hk

GitHub Hugging Face

arXiv:2606.24667v1 [cs.CL] 23 Jun 2026

Abstract

Dense retrieval embedding models are a fundamental component of modern retrieval-based AI systems. Most dense retrievers are trained with contrastive objectives, which require labeled positive and negative document pairs that are often costly and difficult to obtain. In this work, we investigate whether the autoregressive next-token prediction objective of a large language model (LLM) can provide supervision for dense retrieval. The intuition is simple: if a document contains information relevant to a query, conditioning on that document should make the target output easier for the LLM to predict. A key challenge is that the next-token prediction loss is computed inside the LLM, while the retriever is a separate embedding model. To address this challenge, we propose **DREAM** (**D**ense **R**etrieval **E**mbeddings via **A**utoregressive **M**odeling), which injects retriever-generated query-document similarity scores into selected attention heads of a frozen LLM. During training, these scores determine how much attention each candidate document receives while the LLM predicts the target output. The resulting prediction loss provides gradients for retriever training through the attention mechanism. We evaluate DREAM on retrieval benchmarks BEIR and RTEB using embedding backbones ranging from 0.5B to 3B parameters. DREAM consistently outperforms existing baselines across different model scales. These results demonstrate that DREAM provides a promising approach for training dense retrievers through autoregressive modeling.

1 Introduction

Large language model (LLM) systems increasingly rely on retrieval during generation. Retrieval-augmented generation adds external documents to the prompt, and agentic search systems allow an LLM to issue queries, revisit memory, call tools, and plan subsequent actions (Lewis et al., 2020; Wang et al., 2024a; Ni et al., 2026). In these settings, the retriever determines which context is available to the LLM, making retriever training an important part of the overall system.

Most dense retrievers are trained with contrastive objectives. A training example pairs a query with positive documents, which are treated as desired retrieval targets, and sampled negatives, which are treated as lower-relevance alternatives. The objective increases similarity to the positives and decreases similarity to the negatives (Reimers and Gurevych, 2019; Gao et al., 2021; Wang et al., 2024b). In practice, however, constructing positive and negative examples is often the bottleneck. Positive examples typically require expensive relevance annotations, while hard negatives are difficult to mine reliably and may include false negatives (Xiong et al., 2021; Wang et al., 2024c).At the same time, autoregressive modeling via next-token prediction (NTP) has become the foundation of modern language-model training (Radford and Narasimhan, 2018; Radford et al., 2019). Its success suggests that rich supervision can emerge from predicting future tokens in raw text, without any manually specified relevance labels. This observation raises an intriguing question: *can dense retrievers be effectively trained from the autoregressive next-token prediction objective?*

Next-token prediction could potentially provide a useful supervision signal for retrieval. Retrieved documents are ultimately consumed by an LLM to support generation, and their value lies in how much they help the model produce the desired output. If a candidate document contains useful information for answering a query, conditioning on that document should make the target output easier to predict and reduce the next-token prediction loss. Conversely, documents that do not provide useful information should contribute little to prediction and therefore offer little reduction in loss.

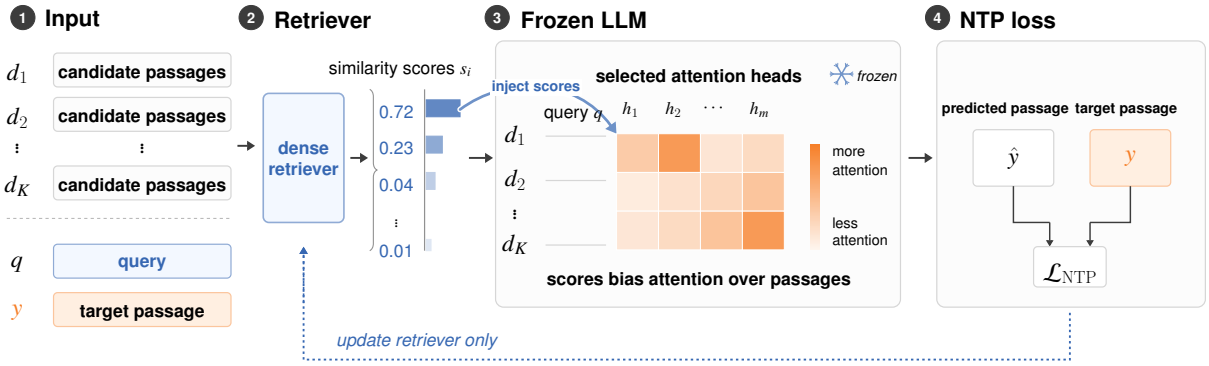

Figure 1: Overview of DREAM. The retriever scores candidate documents for a query, and these scores reweight selected attention heads of a frozen LLM as it predicts the target passage. The next-token prediction loss trains the retriever while the LLM stays frozen.

This gives a direct way to assess retrieval quality: measure its effect on the LLM's prediction loss.

However, translating this signal into a training objective for a dense retriever is not straightforward. The autoregressive NTP loss is computed within the LLM, while the retriever is a separate embedding model that ranks documents using query-document similarity scores. If these similarity scores do not influence the LLM's computation, the loss provides no gradient signal for updating the retriever. The key challenge is therefore to connect the retriever's similarity scores to the LLM computation during training, while keeping the resulting retriever a standalone embedding model at inference time.

Recent work has begun to explore next-token prediction as a supervision signal for retrieval. RePlug distills document preferences from frozen-LLM likelihoods, but the retriever remains external to the LLM computation (Shi et al., 2024). Revela incorporates retriever-computed similarities into language modeling through in-batch attention, but it trains the language model together with the retriever on raw text batches, without an explicit query or target output in the objective (Cai et al., 2026). As a result, the objective does not directly measure whether a candidate document helps the model produce the desired response. While these methods demonstrate the potential of next-token prediction for retrieval, it remains unclear how to fully exploit autoregressive language modeling as a supervision signal for training dense retrievers.

In this work, we propose DREAM, a method that connects a dense retriever to a frozen LLM through attention, enabling next-token prediction loss to directly supervise retrieval training. The central idea is to use attention as the interface between retrieval and LLM generation: retriever scores determine how much attention is assigned from the query to each candidate document. For each training instance, we concatenate the candidate documents, query, and target output. The retriever computes a similarity score for each query-document pair, and these scores are injected into selected attention heads of the frozen LLM. As a result, the retriever influences the attention assigned from the query to each candidate document, while the next-token prediction loss provides supervision for updating the retriever. The LLM itself remains frozen throughout training, and the resulting retriever can be used as a standalone embedding model at inference time.

This design is motivated by two considerations. First, not all attention heads are equally relevant for retrieval. Many heads specialize in functions unrelated to query-document matching and therefore provide little information about document relevance. We therefore inject retriever scores only into query-focused retrieval heads identified by Zhang et al. (2025a). Because these heads already attend from the query to potentially useful context, modulating their attention allows the next-token prediction loss to provide a more informative supervision signal for retrieval. Second, retriever scores are normalized across candidate documents, creating competition for attention. Because candidates share a fixed attention budget, increasing the weight of one document necessarily reduces the weights of others. This allows the prediction loss to implicitly suppress documents that do not help predict the target output, eliminating the need for explicitly constructed negative examples.

We evaluate the learned retrievers on BEIR

(Thakur et al., 2021) and RTEB (Muennighoff et al., 2023) using embedding backbones ranging from 0.5B to 3B parameters. Across all tested scales, DREAM consistently outperforms RePlug and Revela, with gains ranging from 0.015 to 0.081 NDCG@10 on BEIR and from 0.068 to 0.102 on RTEB. These results demonstrate that next-token prediction can serve as an effective supervision signal for dense retrieval. Our analysis further explains why the approach works. We find that the supervision signal is most effective when retrieval scores are injected into attention heads that already gather evidence for the query. Since the LLM remains frozen, these retrieval heads provide a natural interface through which prediction loss can guide retrieval. In contrast, injecting retrieval scores into randomly chosen heads leads to substantially weaker performance. Overall, the experimental results suggest that autoregressive next-token prediction is a viable alternative to contrastive supervision for training dense retrievers.

2 Related Work

This work is related to three areas: retrieval-model training, language-model supervision for retrieval, and attention heads in language models.

Retrieval models. Most retrieval models learn to map queries and documents into a shared space where a query lies close to its relevant documents. Dense dual-encoders embed a query and a passage separately and rank them by similarity (Karpukhin et al., 2020), and this contrastive recipe has been strengthened with large-scale labeled data (Nguyen et al., 2016) and hard-negative mining (Xiong et al., 2021; Qu et al., 2021). Sentence-BERT and SimCSE bring the same matching idea to sentence embeddings (Reimers and Gurevych, 2019; Gao et al., 2021), and E5 scales weakly supervised text-pair training (Wang et al., 2022). More recently, decoder-only LLMs serve as embedding backbones and as generators of synthetic query-document pairs when labels are scarce (Bonifacio et al., 2022; Wang et al., 2024b). Across these architectures, the training signal is still a pairwise matching objective over labeled or synthesized pairs. Constructing such pairs can be costly, and the resulting supervision may be noisy due to annotation errors or false negatives. DREAM takes a different approach by deriving supervision directly from the next-token prediction objective of a frozen LLM rather than from pre-constructed positive and negative pairs.

Language-model supervision for retrieval. Another line of work uses a language model to supervise retrieval. REALM jointly trains a retriever with a masked language model so that the language-model objective shapes retrieval (Guu et al., 2020). RePlug trains a retriever from frozen-LLM likelihoods but keeps the retriever scores outside the LLM forward pass (Shi et al., 2024). Revela trains dense retrievers with a language-modeling objective over chunk sequences, jointly updating the retriever and the language model (Cai et al., 2026). These methods show that language-model feedback can supervise retrieval. However, RePlug and Revela do not directly train the retriever in the query-candidate-target setting used at inference. RePlug distills preferences from frozen-LLM likelihoods, and Revela optimizes sequential chunk prediction over raw text batches. In both cases, the retriever is not directly supervised by whether a candidate document helps the LLM produce the target output for a given query. DREAM addresses this gap by injecting retriever scores into the frozen LLM computation, so the next-token prediction loss trains the retriever through the documents it weights.

Attention heads in language models. Multi-head attention lets different heads specialize in distinct token-to-token computations (Vaswani et al., 2017). Interpretability work shows that induction heads implement structured copying that underlies in-context learning (Olsson et al., 2022), and more recent work identifies query-focused retrieval heads whose attention links query tokens to the relevant parts of a long context (Zhang et al., 2025a). Prior work mainly uses retrieval heads to analyze how LLMs route information. DREAM instead uses these heads for dense retriever training.

3 Method

Each training instance contains a query q, candidate documents $d_1, \ldots, d_K$, and a target passage y. DREAM trains a dense retriever by feeding the retriever's query-document scores into selected attention heads of a frozen decoder-only LLM as it predicts the target passage. This frozen LLM acts as a judge that evaluates how well the retriever's selected documents help predict the target output. Fig. 1 summarizes the training architecture.

3.1 Query-Document Similarity Scores

The trainable model is a dense retriever f_ϕ. Given the query and each candidate document, the re-

triever produces L2-normalized last-token representations:

$$e_q = \frac{f_\phi(q)}{\|f_\phi(q)\|_2}, \qquad e_{d_j} = \frac{f_\phi(d_j)}{\|f_\phi(d_j)\|_2}. \quad (1)$$

The query-document similarity score is

$$s_\phi(q, d_j) = e_q^\top e_{d_j}. \quad (2)$$

This is the score used by the final retriever at inference time. During training, we normalize the candidate scores into document-level weights:

$$p_\phi(d_j \mid q) = \frac{\exp(s_\phi(q, d_j)/\tau)}{\sum_{k=1}^{K} \exp(s_\phi(q, d_k)/\tau)}, \quad (3)$$

where τ is a learnable temperature. The distribution $p_\phi(d_j \mid q)$ is the document-level signal that enters the frozen LLM during training.

3.2 Selecting Query-Focused Heads

We select the attention heads before training, using the query-focused retrieval-head procedure of Zhang et al. (2025a). The goal is to find heads that already perform a retrieval function: their query-token attention assigns higher weight to candidate documents that support the target. We inject retriever scores into these heads because their attention is already organized around query-document relevance. This makes the loss sensitive to whether the retriever gives higher scores to useful documents. Injecting scores into unrelated heads would instead perturb computations that are not about retrieval and provide a noise training signal.

Each probe example follows the same query-candidate-target format as training. The candidate chunk that supports the target passage is treated as the relevant document for head selection. We place the candidate documents before the query, as in the training input. Let D_j be the token span of candidate document d_j, and let $Q(q)$ be the query-token positions. For each head h, we measure how much attention flows from the query tokens to each candidate document:

$$r_h(q, d_j) = \frac{1}{|Q(q)|} \sum_{a \in Q(q)} \sum_{b \in D_j} A_h^q(a, b), \quad (4)$$

where $A_h^q(a, b)$ is the post-softmax attention weight in the frozen LLM. This raw score can reflect position or prompt-format bias, not only query-specific evidence use. We therefore compute a query-independent baseline by replacing the query with a content-free query q_{null}, such as `N/A`, and subtract it:

$$\tilde{r}_h(q, d_j) = r_h(q, d_j) - r_h(q_{\text{null}}, d_j). \quad (5)$$

For each head, we rank the candidate documents by $\tilde{r}_h(q, d_j)$, compute NDCG@10 against the known relevant documents, and average over the probe set. We keep the top M heads and denote them by H. Only these heads receive the retriever-guided attention in the next step.

3.3 Injecting Scores into Attention

During training, the candidate documents, query, and target passage are concatenated into a single decoder-only LLM input. We use document-first causal order:

$$\begin{aligned} \mathcal{I} = (&\textsc{Passages}: d_1, \ldots, d_K \\ &\textsc{Question}: q \\ &\textsc{Target passage}: y). \end{aligned} \quad (6)$$

The candidate documents come first because the query and target tokens can only attend to earlier positions. We use the same D_j and $Q(q)$ notation as above. This order lets the query read the candidate documents and lets the target passage read the query. We therefore inject scores only into query-token rows. The scores change which candidate documents the query gathers evidence from, and the target loss evaluates that evidence when predicting y.

For a selected head $h \in H$ and query-token row $a \in Q(q)$, DREAM separates attention into two choices: which document to read and which tokens to read inside that document. The retriever should control the first choice, while the frozen LLM should keep the second choice. Let $\alpha_h(a, b)$ be the original attention from a to token b. To keep only the LLM's token preference inside document d_j, we normalize the attention over the tokens in D_j:

$$\hat{\alpha}_{h,j}(a, b) = \frac{\alpha_h(a, b)}{\sum_{b' \in D_j} \alpha_h(a, b')}, \qquad b \in D_j. \quad (7)$$

This normalized distribution sums to one inside d_j. We then multiply it by the retriever weight $p_\phi(d_j \mid q)$, so the document receives the total mass chosen by the retriever and distributes that mass across its tokens according to the frozen LLM. For $b \in D_j$,

$$\alpha_h^R(a, b) = p_\phi(d_j \mid q)\, \hat{\alpha}_{h,j}(a, b). \quad (8)$$

For tokens outside the candidate document spans, $\alpha_h^R(a,b) = 0$. Thus, $p_\phi(d_j \mid q)$ controls attention across documents, and $\hat{\alpha}_{h,j}(a,b)$ controls attention within each document.

Finally, the selected head mixes the original attention with this score-guided attention:

$$\alpha'_h(a,b) = (1-g)\,\alpha_h(a,b) + g\,\alpha_h^R(a,b), \quad (9)$$

where $g = \sigma(\gamma) \in [0,1]$ is a learnable gate. We apply this mixture only to selected heads and query-token rows, all other attention rows keep the original attention. The gate controls how strongly the retriever's document choice changes the frozen head.

3.4 Training Objective

The modified attention is used only during training. The frozen LLM predicts the target passage, and the loss is standard next-token cross entropy on the target-passage tokens.

$$\mathcal{L}_{\text{NTP}} = -\sum_{t \in \mathcal{T}_Y} \log p_\theta(x_t \mid x_{<t}; \{\alpha'_h\}_{h \in H}), \quad (10)$$

where θ denotes frozen LLM parameters and $\mathcal{T}_Y$ indexes the target-passage tokens. The gradients do not update θ. They flow from the target-passage loss through α'_h, into $p_\phi(d_j \mid q)$, through the similarity scores $s_\phi(q, d_j)$, and finally into the retriever. If a candidate document helps the frozen LLM predict the target passage, increasing its similarity score can reduce the loss. If it does not help, increasing its score is penalized by the same loss.

This objective differs from likelihood distillation in how the supervision reaches the retriever. The next-token prediction loss directly updates the retriever through the modified attention, rather than first producing judge scores for the retriever to imitate. As a result, a score change is useful only when it helps the frozen LLM predict the target passage. The objective also creates competition among candidates without mined negatives. Because $p_\phi(\cdot \mid q)$ is normalized over the candidate set, increasing the weight of one document lowers the weights of the others. Gradients from the prediction loss therefore move attention toward candidates that help predict the target and away from candidates that do not.

4 Experiments

We evaluate whether DREAM can train a stronger standalone retriever. The main experiment compares retrieval performance against lexical and LLM-supervised retrieval baselines, and Section 5 then examines why the next-token prediction signal works and which design choices it depends on.

4.1 Experimental Setup

Training data. We build the training data from the Wikipedia corpus[1]. Each document is split into 16 chunks, forming one candidate chunk set. For each set, we choose one chunk as the target passage and ask Qwen3-14B (Yang et al., 2025) to generate a query whose answer is supported by that chunk. The target chunk is used as the positive retrieval target, while the full chunk set provides the candidate documents seen during training. The query-generation prompt is provided in Section A.

Implementation. DREAM uses a frozen Llama-3.1-8B-Instruct (Grattafiori et al., 2024) as the next-token prediction judge. Unless otherwise stated, we use the top 16 heads from the query-focused retrieval-head ranking, listed in Section B. We train LoRA adapters on the embedding model's $q/k/v/o$ projection modules with rank 32 and alpha 64. Training uses learning rate 10^{-4}, gradient accumulation 32, batch size 1, 1500 steps, and 16 candidate documents per sample.

Baselines. The main baselines are BM25, RePlug, and Revela (Shi et al., 2024; Cai et al., 2026). BM25 is a lexical retrieval baseline and does not use a learned embedding backbone. InfoNCE (Oord et al., 2018) is a contrastive baseline trained on the same data and candidate pool as DREAM, using the target chunk as the positive and the other candidates in the same set as negatives. To keep the comparison controlled, we do not apply hard-negative mining, so InfoNCE and DREAM differ only in the training objective. RePlug is a frozen-LLM likelihood distillation baseline, where the retriever learns from document preferences induced by LLM likelihoods. Revela is an LLM-supervised dense retrieval baseline that trains from language-modeling signals over text chunks. RePlug, Revela, and DREAM are compared at 0.5B, 1B, and 3B embedding scales. The three scales use Qwen2.5-0.5B (Qwen et al., 2025), Llama-3.2-1B, and Llama-3.2-3B (Grattafiori et al., 2024) as the embedding backbones.

Benchmarks. We evaluate with NDCG@10 on BEIR (Thakur et al., 2021) and RTEB (Muen-

[1] https://huggingface.co/datasets/Tevatron/wikipedia-nq-corpus

nighoff et al., 2023). BEIR covers nine retrieval tasks spanning argument, biomedical, financial, scientific, and community-question-answering domains. RTEB covers fourteen retrieval tasks spanning legal, financial, code, structured-data, and medical domains. Detailed per-task scores are provided in Section C.

4.2 Main Retrieval Results

Best average retrieval performance. Table 1 tests whether the next-token prediction signal produces a stronger retriever under the same embedding backbone. For BM25, DREAM is slightly better on BEIR at 0.5B and clearly better at larger scales, and it is better on RTEB at all scales. DREAM also surpasses InfoNCE at every scale. Since InfoNCE and DREAM use the same data and candidate pool, this gap points to the training objective rather than the shared candidate set. DREAM further outperforms RePlug and Revela, with gains over Revela ranging from 0.015 to 0.081 NDCG@10 on BEIR and from 0.068 to 0.102 on RTEB.

The detailed results in Tables 4 and 5 show that these gains are not driven by a single dataset. On BEIR, DREAM becomes strongest on nearly all tasks as the backbone grows, including scientific, biomedical, and community-question-answering tasks. On RTEB, the improvements are also broad, with especially large gains on code and structured-data retrieval tasks such as Apps, MBPP, and WikiSQL. This suggests that connecting retrieval scores to selected LLM attention heads gives a stronger and transferable training signal.

5 Analysis: Why Does the Signal Work?

The main results show that the training signal works. This section asks why.

5.1 The Interface Determines the Signal

Effect of head selection. The head-selection analysis tests whether retrieval scores can be inserted into any attention head, or whether they must enter heads whose original attention already follows query-candidate relevance. Figure 2 shows that the interface matters. Fully random heads reach only 0.0637 average BEIR NDCG@10 and 0.0320 average RTEB NDCG@10, while selected heads reach 0.4888 and 0.5514. Random middle-layer heads improve over fully random heads, but remain far below the selected heads. This follows

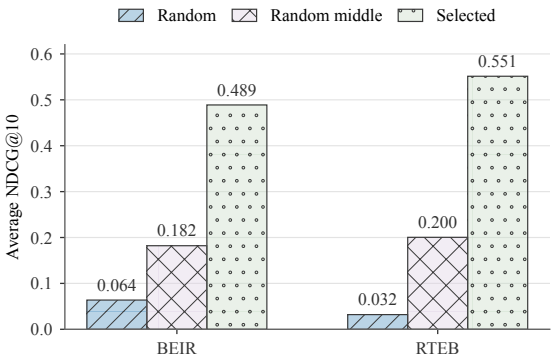

Figure 2: Head selection analysis on BEIR and RTEB with the Llama-3.2-1B backbone. Heads selected from the frozen LLM's query-focused retrieval-head ranking provide much stronger supervision than random heads or random middle-layer heads.

from how training works: because the LLM is frozen, DREAM cannot make arbitrary heads learn a new retrieval role. The retrieval scores are useful only when they modulate heads that already affect how the query reads candidate context. In those heads, increasing the weight of a useful candidate can lower the next-token prediction loss, so the loss guides the retriever on which candidates to rank higher. In unrelated heads, the same score changes disturb computations that are not organized around query-candidate relevance, so the resulting gradients are weak or noisy for retrieval.

Number of selected heads. Figure 3 shows a clear trend under the same Llama-3.2-1B setting. Performance improves as the number of selected heads increases from Top 1 to Top 16, then drops when the set expands to Top 32 and Top 64. This suggests that one head is not enough to carry the training signal, while adding too many heads can include weaker retrieval heads and dilute the signal. Top 16 provides the best balance in this experiment.

5.2 What the Retriever Learns

Embedding geometry. We use two embedding-space metrics to understand what the retriever learns beyond retrieval scores. Alignment measures the average squared distance between a query embedding and its positive-document embedding. Uniformity measures how spread out the query and positive-document embeddings are in the representation space (Gao et al., 2021). Lower alignment means positives are closer to their queries, while lower uniformity means the representations are less collapsed. We compute these metrics on 5,000

Method	BEIR			RTEB		
	Qwen2.5 0.5B	Llama-3.2 1B	Llama-3.2 3B	Qwen2.5 0.5B	Llama-3.2 1B	Llama-3.2 3B
BM25		0.4122			0.3176	
InfoNCE	0.2993	0.3268	0.3339	0.2950	0.3658	0.3405
REPLUG	0.2593	0.2535	0.2705	0.2782	0.2855	0.3250
REVELA	0.4011	0.4075	0.4315	0.4107	0.4499	0.4945
DREAM	**0.4163**	**0.4888**	**0.5074**	**0.4788**	**0.5514**	**0.5892**

Table 1: Average NDCG@10 on BEIR and RTEB. Each benchmark block reports three embedding backbones (Qwen2.5-0.5B, Llama-3.2-1B, Llama-3.2-3B). BM25 is a lexical baseline with no embedding backbone, so its score is shown once per benchmark.

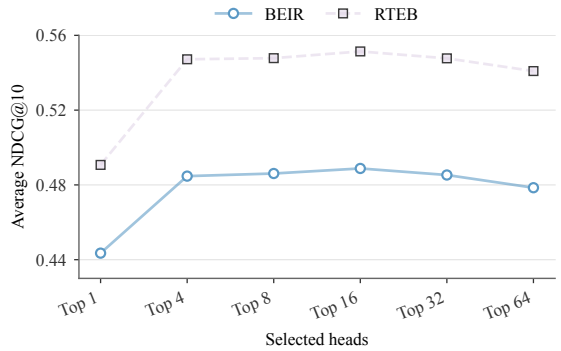

Figure 3: Average NDCG@10 when varying the number of selected heads from the query-focused retrieval-head ranking.

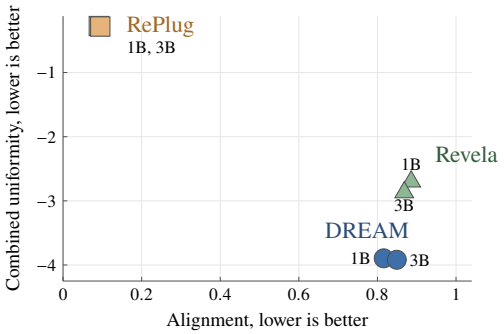

Figure 4: Embedding-space analysis on 5,000 query-positive pairs. Lower is better on both axes.

query-positive pairs randomly sampled from the MTEB retrieval data.

Figure 4 reports the results for the 1B and 3B embedding backbones. RePlug places positive documents very close to their queries, but the embeddings are poorly spread out. Revela improves the spread of the space, but with weaker alignment. DREAM achieves the best uniformity while keeping alignment in the same range as Revela. The gain is not just better alignment. The retriever also learns a less collapsed embedding space. This better uniformity likely comes from competition among candidates during training. The document weights sum to one over each candidate set, so the retriever is rewarded for pushing unhelpful documents away from the query, not only for pulling the helpful one closer. This embedding geometry is particularly encouraging because DREAM is not trained with a contrastive objective, which is typically associated with improving alignment and uniformity. Despite relying solely on autoregressive next-token prediction, DREAM learns representations with similar geometric properties.

6 Training Ablations

Effect of updating the judge LLM. Figure 5 compares the main frozen-LLM setting with a variant that also adds LoRA adapters to the judge LLM during training. Keeping the judge frozen is better in all four comparisons. This suggests that the fixed LLM computation provides a more stable training signal for the retriever, while updating the judge can weaken the signal that reaches the retrieval model. This result follows from the role of the judge in our objective. The frozen LLM acts as a fixed judge of document usefulness: if the retriever gives more weight to documents that help predict the target, the loss should fall. When the LLM is also updated, the loss can fall for another reason: the LLM changes its own prediction behavior. Then a lower loss no longer points as directly to better document weights, so the gradient gives the retriever a weaker training signal. This is why freezing the LLM is useful in our setting. The model that judges document usefulness stays

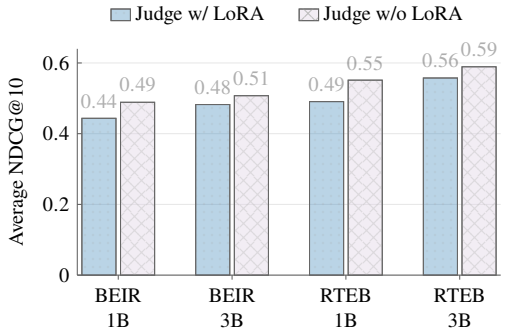

Figure 5: Effect of updating the judge LLM. The frozen-judge setting trains only the embedding model adapters, while the judge LoRA variant also updates LoRA adapters in the LLM.

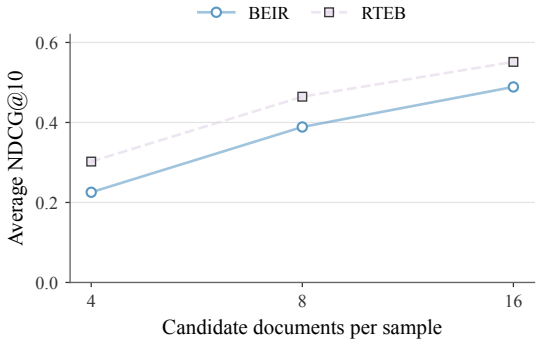

Figure 6: Effect of the number of candidate documents per training sample with the Llama-3.2-1B embedding backbone.

fixed, and only the retriever learns to change which documents it lets the LLM read.

Number of candidate documents. Figure 6 varies the number of candidate documents in each training sample using the Llama-3.2-1B embedding backbone. Increasing the candidate set from 4 to 16 improves average NDCG@10 on both BEIR and RTEB, which suggests that the training loss benefits from having enough alternatives to compare. This fits the competition view of the objective: a larger candidate set gives each training step more documents to compare, which sharpens the signal. We therefore use 16 candidate documents as the default setting.

Scaling to larger models. We test whether the same training recipe remains useful with an 8B-scale embedding backbone. Specifically, we train DREAM-8B from Llama-3.1-8B (Grattafiori et al., 2024) using the same recipe. Table 2 compares DREAM-8B with several strong off-the-shelf embedding models, including BGE-large-en-v1.5 (Xiao et al., 2024), Cohere-embed-english-v3.0[2], E5-mistral-7b-instruct (Wang et al., 2024b), and Qwen3-Embedding-8B (Zhang et al., 2025b). DREAM-8B reaches 0.5531 average NDCG@10 on BEIR and 0.6417 on RTEB, matching E5-mistral-7b-instruct on BEIR and achieving a 0.0386 higher score on RTEB. While DREAM-8B does not surpass Qwen3-Embedding-8B, this comparison should be interpreted with caution, as Qwen3-Embedding-8B is trained on a different backbone model with a more carefully curated datasets. Our goal in this experiment is not to establish a new state of the art, but to evaluate whether autoregressive next-token prediction remains an effective supervision signal at larger model scales. The strong performance of DREAM-8B suggests that the proposed training paradigm scales favorably and continues to produce competitive retrieval models.

Model	BEIR	RTEB
BGE-large-en-v1.5	0.5360	0.4943
Cohere-embed-english-v3.0	0.5406	0.5083
E5-mistral-7b-instruct	0.5526	0.6031
DREAM-8B (ours)	0.5531	0.6417
Qwen3-Embedding-8B	0.6348	0.7383

Table 2: Average NDCG@10 with an 8B-scale embedding model. The comparison is included as a scaling check.

7 Conclusion

We introduced DREAM, a method for training standalone dense retrievers through autoregressive next-token prediction. By injecting query-document scores into selected attention heads of a frozen LLM, DREAM enables retrieval training without manually labeled relevance pairs. Across BEIR and RTEB, DREAM consistently outperforms existing LLM-supervised retrieval baselines. Our analyses further show that this supervision signal works best through query-focused retrieval heads and naturally produces a better embedding space. Overall, our results suggest that autoregressive next-token prediction is a practical and scalable alternative to contrastive supervision for training retrievers.

[2] https://cohere.com/blog/introducing-embed-v3

References

Luiz Henrique Bonifacio, Hugo Queiroz Abonizio, Marzieh Fadaee, and Rodrigo Nogueira. 2022. Inpars: Data augmentation for information retrieval using large language models. *CoRR*, abs/2202.05144.

Fengyu Cai, Tong Chen, Xinran Zhao, Sihao Chen, Hongming Zhang, Tongshuang Wu, Iryna Gurevych, and Heinz Koeppl. 2026. Revela: Dense retriever learning via language modeling. In *The Fourteenth International Conference on Learning Representations*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pages 6894–6910.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. 2020. Retrieval augmented language model pre-training. In *International conference on machine learning*, pages 3929–3938. PMLR.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pages 6769–6781.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2014–2037.

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. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, CEUR Workshop Proceedings. CEUR-WS.org.

Xiaoqi Ni, Jie Wang, Lin Yang, Yiyang Lu, Hanzhu Chen, Rui Liu, and Jianye HAO. 2026. Following the navigation: Enhancing small language models contextual reasoning with LLM guidance. In *The Fourteenth International Conference on Learning Representations*.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, et al. 2022. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*.

Yingqi Qu, Yuchen Ding, Jing Liu, Kai Liu, Ruiyang Ren, Wayne Xin Zhao, Daxiang Dong, Hua Wu, and Haifeng Wang. 2021. Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, pages 5835–5847. Association for Computational Linguistics.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei

Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report.

Alec Radford and Karthik Narasimhan. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. 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 3982–3992.

Weijia Shi, Sewon Min, Michihiro Yasunaga, Minjoon Seo, Richard James, Mike Lewis, Luke Zettlemoyer, and Wen-tau Yih. 2024. Replug: Retrieval-augmented black-box language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 8371–8384.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. 2024a. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *CoRR*, abs/2212.03533.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024b. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11897–11916.

Shiqi Wang, Yeqin Zhang, and Cam-Tu Nguyen. 2024c. Mitigating the impact of false negative in dense retrieval with contrastive confidence regularization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19171–19179.

Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. 2024. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th international ACM SIGIR conference on research and development in information retrieval*, pages 641–649.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N. Bennett, Junaid Ahmed, and Arnold Overwijk. 2021. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

Wuwei Zhang, Fangcong Yin, Howard Yen, Danqi Chen, and Xi Ye. 2025a. Query-focused retrieval heads improve long-context reasoning and re-ranking. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 23802–23816.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, et al. 2025b. Qwen3 embedding: Advancing text embedding and reranking through foundation models. *arXiv preprint arXiv:2506.05176*.

A Query-Generation Prompt

We use the following prompt to generate a query from the target passage in each candidate chunk set.

```
Read the following passage and write ONE question that can ONLY be answered from this passage.

Requirements:
- The answer must come from information in this passage.
- Do NOT ask about general knowledge outside the passage.
- The question should require reading more than one sentence to answer.

Output ONLY the question inside the XML tags:

<question>
Your question here
</question>

Passage:
{target_passage}

Question:
```

B Selected Attention Heads

The default DREAM setting uses the top 16 attention heads from the query-focused retrieval-head ranking of the frozen Llama-3.1-8B-Instruct judge. The ranking is computed on 5,000 examples from the training data before retriever training. Table 3 lists the selected heads. Layer and head indices are zero-based, following the indexing used in the detection code.

Rank	Head (ℓ, h)	Rank	Head (ℓ, h)
1	(13, 18)	9	(16, 19)
2	(14, 22)	10	(17, 26)
3	(14, 13)	11	(16, 1)
4	(14, 31)	12	(16, 8)
5	(14, 20)	13	(20, 1)
6	(13, 1)	14	(24, 27)
7	(13, 13)	15	(16, 25)
8	(17, 24)	16	(13, 21)

Table 3: Top 16 selected attention heads used by default in DREAM. Each head is shown as (layer, head).

C Per-Task Retrieval Results

Tables 4 and 5 report per-task NDCG@10 for the main experiment. These tables supplement the average scores in Table 1.

Method	Avg.	ArguAna	NFCorpus	FiQA	SciFact	SCIDOCS	Quora	TREC-COVID	Touche	CQADupStack
BM25	0.4122	0.4629	0.3098	0.2339	0.6644	0.1503	0.8067	0.6504	0.2493	0.1818
Base model: Qwen2.5-0.5B										
INFONCE	0.2993	0.3670	0.1027	0.1748	0.4590	0.0473	0.8483	0.3455	0.1186	0.2307
REPLUG	0.2593	0.3350	0.0543	0.1617	0.3164	0.0111	0.8083	0.4191	0.0891	0.1389
REVELA	0.4011	0.4262	0.2365	**0.2830**	**0.6434**	**0.1467**	0.8394	0.4808	**0.1930**	0.3610
Ours	**0.4163**	**0.5637**	**0.2522**	0.2756	0.6343	0.1376	**0.8617**	**0.5233**	0.1017	**0.3964**
Base model: Llama-3.2-1B										
INFONCE	0.3268	0.4416	0.1033	0.2044	0.5659	0.0662	0.8586	0.3860	0.0642	0.2512
REPLUG	0.2535	0.3081	0.0450	0.1776	0.3138	0.0113	0.8221	0.4068	0.0686	0.1287
REVELA	0.4075	0.4543	0.2587	0.3208	0.7018	0.1726	0.8314	0.4258	**0.1733**	0.3287
Ours	**0.4888**	**0.5762**	**0.3532**	**0.3907**	**0.7243**	**0.1958**	**0.8691**	**0.6983**	0.1572	**0.4343**
Base model: Llama-3.2-3B										
INFONCE	0.3339	0.4462	0.1475	0.1912	0.5984	0.0435	0.8449	0.4158	0.0562	0.2613
REPLUG	0.2705	0.3726	0.0700	0.1776	0.3718	0.0131	0.8278	0.3711	0.0581	0.1724
REVELA	0.4315	0.4794	0.3146	0.3503	0.7192	0.1796	0.8298	0.4860	0.1449	0.3800
Ours	**0.5074**	**0.5765**	**0.3806**	**0.4538**	**0.7565**	**0.2135**	**0.8692**	**0.6734**	**0.1614**	**0.4815**

Table 4: BEIR per-task NDCG@10. Results are grouped by the base embedding model. BM25 has no learned backbone.

Method	Avg.	AILA-C	AILA-S	LegalSum	FinBench	HC3Fin	FinQA	Apps	DS1000	HumanEval	MBPP	WikiSQL	FreshStack	ChatDoctor	CUREv1
BM25	0.3176	0.2932	0.1646	0.5543	0.3160	0.2655	0.7653	0.0104	0.3297	0.3497	0.0919	0.4365	0.2627	0.2749	0.3325
Base model: Qwen2.5-0.5B															
INFONCE	0.2950	0.1407	0.1386	0.5679	0.3106	0.2753	0.2820	0.0317	0.2226	0.5653	0.5465	0.4108	0.0793	0.2024	0.3560
REPLUG	0.2782	0.1530	0.1759	0.3285	0.3360	0.2139	0.4280	0.0238	0.3571	0.4186	0.4614	0.3038	0.1232	0.2026	0.3691
REVELA	0.4107	**0.2975**	0.1595	**0.5780**	0.3054	0.3752	**0.4579**	0.1054	0.4585	0.7950	0.7489	0.5327	0.1956	0.3433	**0.3969**
Ours	**0.4788**	0.1957	**0.2428**	0.5145	**0.4639**	**0.3812**	0.3974	**0.2656**	**0.5523**	**0.9005**	**0.8170**	**0.9176**	**0.2445**	**0.4942**	0.3167
Base model: Llama-3.2-1B															
INFONCE	0.3658	0.1618	0.2048	0.5737	0.4821	0.2968	0.4076	0.0593	0.2570	0.6722	0.6006	0.5876	0.1444	0.3019	0.3719
REPLUG	0.2855	0.1460	0.1829	0.4046	0.4700	0.2326	0.4722	0.0197	0.2749	0.3052	0.2797	0.4192	0.1565	0.2356	0.3975
REVELA	0.4499	**0.2764**	0.2062	**0.6037**	0.5071	0.4201	**0.5456**	0.1587	0.5447	0.7661	0.7137	0.5897	0.2477	0.3380	0.3809
Ours	**0.5514**	0.2420	**0.3078**	0.5945	**0.7095**	**0.5402**	0.4874	**0.2530**	**0.5888**	**0.8962**	**0.8298**	**0.9451**	**0.2931**	**0.5649**	**0.4679**
Base model: Llama-3.2-3B															
INFONCE	0.3405	0.1863	0.1809	0.5010	0.4794	0.3543	0.3698	0.0272	0.2607	0.6148	0.4141	0.5432	0.1102	0.3248	0.3999
REPLUG	0.3250	0.1622	0.1370	0.4858	0.5541	0.2599	0.4538	0.0354	0.3232	0.6053	0.3020	0.3666	0.1632	0.2741	0.4279
REVELA	0.4945	0.2606	0.2529	**0.6272**	0.6105	0.4878	**0.5369**	0.2055	0.5632	0.8655	0.7698	0.6196	0.2872	0.4093	0.4275
Ours	**0.5892**	**0.2983**	**0.3196**	0.6254	**0.7985**	**0.6457**	0.4768	**0.3860**	**0.6034**	**0.9523**	**0.8708**	**0.8401**	**0.3320**	**0.5998**	**0.5005**

Table 5: RTEB per-task NDCG@10. Results are grouped by the base embedding model. BM25 has no learned backbone.