Following the Autoregressive Nature of LLM Embeddings via Compression and Alignment

Anonymous ACL submission

Abstract

A new trend uses LLMs as dense text encoders via contrastive learning. However, since LLM embeddings predict the probability distribution of the next token, they are inherently generative and distributive, conflicting with contrastive learning, which requires embeddings to capture full-text semantics and align via cosine similarity. This discrepancy hinders the full utilization of LLMs' pre-training capabilities, resulting in inefficient learning. In response to this issue, we propose AutoRegEmbed, a new contrastive learning method built on embedding conditional probability distributions, which integrates two core tasks: information compression and conditional distribution alignment. The information compression task encodes text into the embedding space, ensuring that the embedding vectors capture global semantics. The conditional distribution alignment task focuses on aligning text embeddings with positive samples embeddings by leveraging the conditional distribution of embeddings while simultaneously reducing the likelihood of generating negative samples from text embeddings, thereby achieving embedding alignment and uniformity. Experimental results demonstrate that our method significantly outperforms traditional contrastive learning approaches and achieves performance comparable to state-of-the-art models when using the same amount of data. Our code is available at `https://anonymous.4open.science/r/AutoRegEmbed-7530`

1 Introduction

Text embeddings, which represent the semantic content of natural language text as vectors, are extensively utilized in domains such as information retrieval, semantic similarity assessment, and retrieval-augmented generation (RAG) (Khandelwal et al., 2020; Shi et al., 2024; Deng et al., 2023; Ding et al., 2024). Traditional text embedding models typically employ transformer-based architectures with encoder-only designs, including examples like Bert (Devlin et al., 2019), DeBERTa (He et al., 2021) and MPNet (Song et al., 2020), and are trained using contrastive learning.

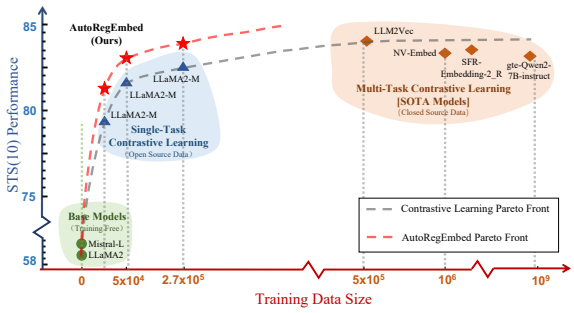

Figure 1: Comparison of pareto front between AutoRegEmbed and other methods. The horizontal axis represents the number of training samples, while the vertical axis indicates the average performance across 10 STS datasets. The upper left corner represents the region with the highest learning efficiency.

Recently, contrastive learning has been directly applied to decoder-only LLMs, which are trained to generate embedding vectors based on task-specific instructions, enabling adaptability to various embedding scenarios (Lee et al., 2024; BehnamGhader et al., 2024). Despite initial advancements, training a high-performance 7B-scale text embedding model using this approach remains highly resource-intensive. It typically requires millions of triplets (Wang et al., 2024a; Li et al., 2024b, 2023) and substantial computational power, including thousands of hours on an A100 80GB GPUs (Muennighoff et al., 2024; Ma et al., 2024), even with the application of Parameter-Efficient Fine-Tuning (PEFT) (Hu et al., 2022; Dao, 2024). The high resource consumption might reasonably be attributed to the inability of the *discriminative* contrastive learning method to fully harness the capabilities of *generative* LLMs (Li et al., 2024a). Firstly, the constraint of unidirectional attention in LLMs leads to the aggregation of information in the hidden state of the output layer corresponding

to the final token. However, as LLMs are optimized for next-token prediction, this hidden state can only represent the semantics of the next token (local) rather than the semantics of the input text itself (global). Consequently, employing this hidden state directly in contrastive learning requires additional training time and computational cost to transition from a localized to a more global semantic representation. Secondly, the hidden state in LLMs is used to generate the probability distribution of the next token, whereas contrastive learning optimizes the cosine distance between the hidden states of different texts. This divergence in optimization objectives introduces additional training costs. This raises an important question: *Is it feasible to develop a method that follows the autoregressive nature while generating high-quality text embeddings and significantly reducing resource requirements?*

We formalize three key requirements to address this problem. Firstly, embeddings should capture global semantics rather than focusing solely on next-token semantics. Secondly, they must follow alignment and uniformity principles (Wang and Isola, 2020). Finally, the transformation from the original embedding to one that meets these criteria should follow an autoregressive nature. To this end, we propose AutoRegEmbed, which encompasses two tasks: information compression and conditional distribution alignment.

The information compression task is inspired by the concept of context compression (Chevalier et al., 2023; Ge et al., 2024; Mu et al., 2023), which addresses the limitations of context window length and the high computational cost faced by LLMs when processing long texts. Specifically, we encode the context and instructions into a set of compressed variables, which are then passed to a decoder with the same architecture but frozen parameters, forcing it to reconstruct the corresponding target. By restricting the decoder to rely solely on the compressed variables—without access to the original context or instructions—we introduce an information bottleneck. This ensures that the compressed variables effectively capture the global semantics of the instructions and context.

The conditional distribution alignment task draws inspiration from traditional contrastive learning and LLM alignment techniques (Wang et al., 2024b). We begin by treating the compressed vectors as embeddings of their corresponding inputs. Then, we adopt the structure of the InfoNCE (van den Oord et al., 2018) loss function, but redefine the similarity metric. Simply put, we align the distance between the conditional probability distributions of text and positive sample embeddings while increasing the likelihood of text embeddings generating positive samples and decreasing the likelihood of generating negative samples. This approach promotes the alignment and uniformity of compressed variables while maintaining the autoregressive nature.

Experimental results on 10 STS datasets demonstrate that AutoRegEmbed outperforms traditional contrastive learning methods while utilizing the same computational resources, making it a highly efficient and scalable solution. Remarkably, even with a limited number of training samples, AutoRegEmbed achieves performance on par with the current state-of-the-art (SOTA) models, showcasing its superior ability to learn robust and generalizable representations from scarce data. As shown in Figure 1, the Pareto frontier of AutoRegEmbed consistently outperforms traditional contrastive learning methods, demonstrating a more optimal trade-off between computational efficiency and performance. This indicates that AutoRegEmbed achieves superior representation learning while maintaining a balanced resource utilization.

2 Related Works

Text embedding is a technique that maps text data into a numerical vector space, capturing both semantic and contextual features of the text. Research in this area can be divided into three categories based on the underlying model: Early models, LLMs with fine-tuning, and LLMs without fine-tuning.

Early Models Early approaches include SentenceBERT (Reimers and Gurevych, 2019) (supervised) and SimCSE (Gao et al., 2021) (unsupervised), which leverage contrastive learning to generate high-quality text embeddings using small encoder-only models. Another area of focus has been improving the isotropy of embedding spaces. Works such as BERT-flow (Li et al., 2020) (flow models) and BERT-whitening (Su et al., 2021) (linear transformations) address the anisotropic properties of embeddings. Meanwhile, multi-stage contrastive learning (Li et al., 2023; Ni et al., 2022; Wang et al., 2022) has further advanced text embeddings by combining pre-training on large weakly supervised datasets with fine-tuning on smaller

high-quality datasets. Inspired by instruction fine-tuning, recent research (Su et al., 2023; Asai et al., 2023) has shifted toward using text paired with instructions to enhance the generalization and transferability of text embeddings in complex scenarios. However, the performance of methods based on early models is limited, due to their reliance on models with relatively small parameter counts.

LLMs with Fine-Tuning Many studies have focused on transforming LLMs into text embedding models through contrastive learning fine-tuning. RepLLaMA (Ma et al., 2024), for example, follows the DPR (Karpukhin et al., 2020) pipeline, using the hidden state of the last token generated by LLaMA as a text embedding vector and applying contrastive learning fine-tuning. Recognizing that the unidirectional attention mechanism in LLMs may limit text embedding quality, LLM2Vec (BehnamGhader et al., 2024) introduces a bidirectional attention mechanism combined with average pooling to enhance embedding quality. NV-Embed (Lee et al., 2024) takes this further by incorporating an additional Latent Attention Layer to generate pooled embeddings. bge-en-icl (Li et al., 2024b) suggests that retaining the original framework of LLMs and leveraging in-context learning is the optimal approach for generating text embeddings. Some studies (Wang et al., 2024a) even use synthetic data generated by LLMs, rather than real-world data, for fine-tuning and achieve competitive performance on the MTEB leaderboard (Muennighoff et al., 2023). However, these approaches often overlook the fundamental differences between language modeling and contrastive learning, failing to fully leverage the potential of LLMs. More closely related to our work is Llama2Vec (Li et al., 2024a), which proposes two pretext tasks to enable unsupervised adaptation of LLMs, followed by contrastive learning fine-tuning to achieve better performance. In contrast, our approach achieves strong results without any need for contrastive learning fine-tuning, as our task fully exploits the inherent potential of LLMs.

LLMs without Fine-Tuning Several studies have explored methods to transform LLMs into text encoders without fine-tuning. (Liu et al., 2024) proposed using possible trajectory distributions as text representations, achieving effectiveness but at a high computational cost. (Springer et al., 2024) introduced echo embeddings by repeatedly feeding text into autoregressive models, addressing architectural limitations but doubling computational requirements. Other methods focus on prompt adjustments to produce meaningful embeddings. PromptEOL (Jiang et al., 2024) introduced a One-Word Limitation prompt to improve embedding performance, while MetaEOL (Lei et al., 2024) extended this idea by using eight different prompt types to generate multi-view embeddings. GenEOL (Thirukovalluru and Dhingra, 2024) leveraged LLMs to create various sentence transformations that retain their meaning, aggregating the resulting embeddings to enhance the overall sentence representation. Meanwhile, PromptReps (Zhuang et al., 2024) developed a hybrid document retrieval framework leveraging prompts to address challenges in information retrieval tasks. Despite these innovations, these approaches either perform poorly or require multiple inferences to achieve good results. By contrast, our method surpasses these methods with minimal training costs.

3 Method

In this section, we first introduce the preliminary information about the task of text embedding with instructions. We then discuss the information compression, which transitions LLM embeddings from local semantics to global semantics, followed by the conditional distribution alignment, which optimizes the conditional probability distribution of embeddings to ensure alignment and uniformity.

3.1 Preliminary

Text embeddings with instructions can adapt to various downstream tasks. Formally, given a large collection $D = \{d_1, d_2, \ldots, d_N\}$ containing N documents, as well as a text q and an instruction t, the embedding $e_{q,t} = E(q, t)$ generated from q and t can match documents $d \in D$ that are relevant to q, according to t, where E represents the text encoder. Thus, by simply changing the instruction t, the relevance measure can be adapted to different downstream tasks. For example, for dense retrieval tasks, the instruction might be "*find documents that can answer this question,*" while for semantic similarity tasks, the instruction could be "*find sentences that are semantically similar to this text*". Numerous studies have explored various embedding techniques and instruction diversities. Our goal is to identify a simple yet effective way to enable LLMs to generate high-quality embeddings directly from autoregressive framework.

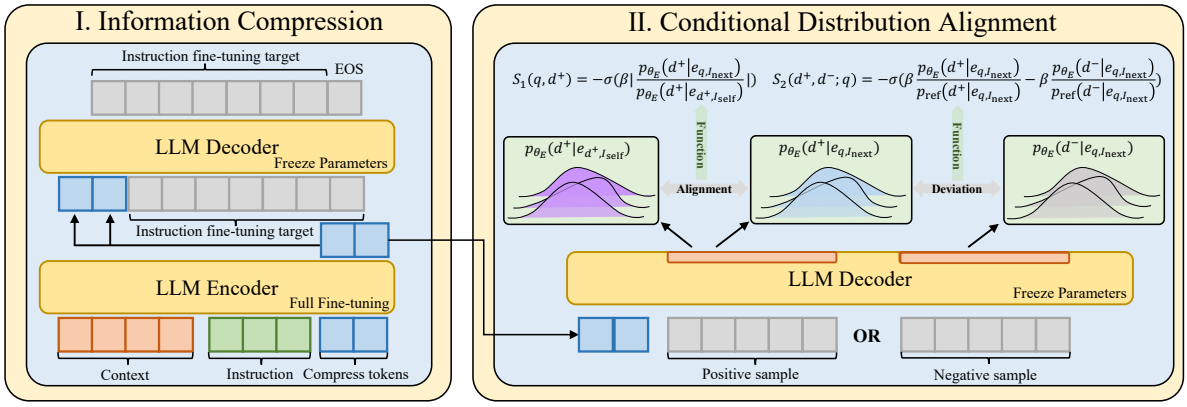

Figure 2: Overall framework of AutoRegEmbed. Firstly, we perform the information compression task to inject key information from the context and instruction into the compressed tokens. Then, we optimize the conditional probability distribution of these tokens to align the distributions of $e_{q,I_{\text{next}}}$ and $e_{d^+,I_{\text{self}}}$ as closely as possible through $S_1(q, d^+)$, while increasing the probability of $e_{q,I_{\text{next}}}$ generating positive samples and reducing the probability of $e_{q,I_{\text{next}}}$ generating negative samples through $S_2(d^+, d^-; q)$.

3.2 Information Compression: from Discriminative to Generative Embeddings

In this section, we first explain the motivation for transitioning from discriminative embeddings to generative embeddings, followed by a formal definition of the information compression task.

In decoder-based LLMs, embeddings are typically generated by extracting the hidden state of the final token in the input sequence. However, this approach primarily captures the semantics of the first output token rather than encoding the global semantics of the entire input. Various pooling techniques, such as average pooling and attention pooling, have been explored to mitigate this limitation, yet they introduce their own challenges. The average pooling method, which computes the mean of all token hidden states, does not necessarily encapsulate global semantics but instead serves as a mechanism for "convexity preservation." Conversely, attention pooling modifies the attention mechanism or introduces additional parameters, thereby altering the original architecture of LLMs. Such modifications deviate from the model's pre-training design and can lead to unintended consequences, as prior studies (Li et al., 2024b) indicate that maintaining the original LLM framework often yields optimal performance. To enable LLMs to generate embeddings that represent global semantics, we introduce an information compression task. This task compels LLMs to reconstruct the original target using a compressed embedding derived from the input text. Given that this compressed embedding models the conditional probability distribution of the target, we designate it as the generative embedding to contrast it with the discriminative embedding produced by conventional pooling approaches.

The information compression task is inspired by the concept of context compression. Specifically, we append k compressed tokens $c = (c_1, \ldots, c_k)$, where $k << n + m$, to the text $q = (q_1, \ldots, q_n)$ and instruction $t = (t_1, \ldots, t_m)$, with n and m representing their respective token lengths. This combined (q, t, c) is then fed into an encoder E to generate the embedding $e_c = (e_{c_1}, \ldots, e_{c_k})$. As mentioned earlier, we expect the embedding e_c to capture the global semantics of the text q and the instruction t. To achieve this, we input e_c into a frozen decoder D, which shares the same architecture, and force it to generate the most relevant document d. The optimization objective for this task can be expressed as:

$$\begin{aligned}\mathcal{L}_{\text{IC}} &= \max_{e_{c_1},\ldots,e_{c_k}} P(d|e_{c_1}, \ldots, e_{c_k}; \theta_D) \\ &= \max_{\Theta_E} P(d|c_1 \ldots c_k, t_1 \ldots t_m, q_1 \ldots q_n; \theta_E, \theta_D),\end{aligned}$$

where θ_E and θ_D denote the parameters of E and D, respectively.

3.3 Conditional Distribution Alignment: from Data-Point to Distribution Perspective

After addressing the global semantic representation issue of the embedding vector, we also require the embedding vector to meet the criteria of alignment and uniformity. Existing studies (Wang and Isola, 2020) have provided specific definitions for these properties. Alignment is typically expressed by the

following formula:

$$\text{Alignment}(f,\alpha) = \mathop{\mathbb{E}}_{(q,d^+)\in p_{\text{pos}}} \parallel f(q) - f(d^+) \parallel_2^{\alpha},$$

where $\alpha > 0$ is a parameter used to adjust the weight of the distance between positive sample pairs (q and d^+), and $p_{\text{pos}}(\cdot,\cdot)$ represents the distribution of positive sample pairs. The smaller $\text{Alignment}(f,\alpha)$ is, the better the alignment of the embedding vector generated by f. Uniformity measures how evenly the embedding is distributed, commonly expressed by the following formula:

$$\text{Uniformity}(f,\alpha) = \log \mathop{\mathbb{E}}_{(q,d)\in p_{\text{data}}} e^{-t\|f(q)-f(d)\|_2^2},$$

where $t > 0$ and p_{data} represents the data distribution. In general, we optimize these two properties asymptotically using a contrastive loss, such as InfoNCE,

$$\mathcal{L}_{\text{InfoNCE}}(f;\tau) = \mathbb{E}[-\log \frac{e^{f(q)^T f(d^+)/\tau}}{e^{f(q)^T f(d^+)/\tau} + \sum_i e^{f(q)^T f(d_i^-)/\tau}}], \tag{1}$$

where τ denotes the temperature parameter and d_i^- represents the i-th negative sample. Clearly, Equation 1 differs fundamentally from the generative pre-training task, as it optimizes the cosine distance between sample embeddings, aligning data points in the embedding space rather than modeling the next-token probability distribution, which is central to pre-training. So, using this loss function to optimize an LLM may not fully unlock its potential.

To address this, we propose the Conditional Distribution Alignment task to minimize this discrepancy as much as possible. The concept is straightforward: Instead of using the cosine distance between embeddings, we assess similarity based on the conditional probability distribution corresponding to each embedding. Simply put, we extend point alignment to distribution alignment. Formally, the decoder L_D is a well-trained autoregressive language model with the following conditional probability distribution:

$$p(d|e_c) = \prod_{t=1}^{T} p(d_t|d_{<t}, e),$$

where $e_c = (e_{c_1}, \ldots, e_{c_k})$ is the embedding variables, $d = (d_1, d_2, \ldots, d_T)$ represents the generated sentence, and $d_{<t}$ denotes the part of the sentence before time step t. Intuitively, the similarity between corresponding samples q and d can be measured by computing the distance between the conditional probability distributions of their embeddings, e_q and e_d:

$$S(q,d) = \frac{1}{T}\sum_{t=1}^{T} \text{D}(p(d_t|d_{<t}, e_q), p(d_t|d_{<t}, e_d)),$$

where $\text{D}(\cdot,\cdot)$ is any function that measures the divergence between two probability distributions. In addition, since the conditional probability distribution of the embedding can be adjusted based on the given instruction, we can compute the logarithmic probability of the text embedding $e_{q,I_{\text{next}}}$ generating positive or negative samples to measure the similarity between the text q and the positive d^+ and negative d^- samples. Here, the instruction I_{next} is similar to "*find documents that can answer this question*", which ensures that the embedding $e_{q,I\text{next}}$ generated from text q produces positive samples after passing through the decoder. For positive and negative samples, we use the instruction I_{self} similar to "*find sentences that are semantically similar to this text*", so that their embeddings, $e_{d^+,I_{\text{self}}}$ and $e_{d^-,I_{\text{self}}}$, generate themselves after passing through the decoder.

Building on the above insights and incorporating the structure of InfoNCE, we empirically derive the final loss function:

$$\begin{aligned}
&\mathcal{L}_{\text{CDA}} = \mathbb{E}[-\log \frac{e^{S_1(q,d^+)/\tau}}{e^{S_1(q,d^+)/\tau} + \sum_i e^{S_2(d^+,d_i^-;q)/\tau}}],\\
&S_1(q,d^+) = -\sigma(\beta\, |\log \frac{p_{\theta_E}(d^+|e_{q,I_{\text{next}}})}{p_{\theta_E}(d^+|e_{d^+,I_{\text{self}}})}|),\\
&S_2(d^+,d_i^-;q) = -\sigma(\beta \log \frac{p_{\theta_E}(d^+|e_{q,I_{\text{next}}})}{p_{\text{ref}}(d^+|e_{q,I_{\text{next}}})}\\
&- \beta \log \frac{p_{\theta_E}(d^-|e_{q,I_{\text{next}}})}{p_{\text{ref}}(d^-|e_{q,I_{\text{next}}})}),
\end{aligned} \tag{2}$$

where τ and β are temperature parameters, and $p_{\Theta_{L_E}}$ represents the initial model. We use the Sigmoid function $\sigma(\cdot)$ to normalize the similarity measured from the conditional probability distribution to the range [0,1], ensuring maximum consistency with the range of cosine distance. S_1 represents the similarity function between text q and the positive sample d^+. We define it by measuring the absolute value of the difference in the logarithmic probability of their corresponding embeddings, $e_{q,I_{\text{next}}}$ and $e_{d^+,I_{\text{self}}}$, generating the positive sample d^+. To minimize this difference, we apply the absolute value function. In addition, we then add a negative

sign to ensure that the value of S_1 increases as the similarity between q and d^+ increases. S_2 calculates the difference between the logarithmic probabilities of generating positive and negative samples for text q, similar to DPO (Rafailov et al., 2023). We amplify this difference to boost the probability of embedding $e_{q,I_{\text{next}}}$ generating positive samples and decrease the probability of generating negative samples. We normalize the probabilities by dividing them by the corresponding values from the initial model to account for the length discrepancy between positive and negative samples.

4 Experiments

4.1 Experimental Settings

Evaluations Previous studies (Gao et al., 2021; Li et al., 2020) highlight that a key goal of text embedding is to cluster semantically similar sentences. Following this approach, we use the MTEB (Muennighoff et al., 2023) evaluation framework to evaluate AutoRegEmbed on ten semantic text similarity datasets, including STS12 (Agirre et al., 2012), STS13 (Agirre et al., 2013), STS14 (Agirre et al., 2014), STS15 (Agirre et al., 2015), STS16 (Agirre et al., 2016), STS17 (Cer et al., 2017), STS22 (Chen et al., 2022), STS-B , BIOSSES and SICK-R. Each pair of text in the STS dataset is labeled with a similarity score ranging from 0 to 5 or 0 to 4, indicating their semantic similarity. The evaluation metric is the Spearman correlation between the similarity scores predicted by the model and the scores annotated by humans.

Training In the information compression stage, we use the training set of the instruction fine-tuning dataset PWC (Ge et al., 2024), which includes a diverse range of instruction types, as the training data. The original dataset contains 241,564 (context, instruction, target) samples. To reduce redundancy caused by repeated contexts, we remove duplicates, resulting in the PWC-Unique dataset with 16,382 samples as the final training data. In the conditional distribution alignment stage, we use NLI data (Wang et al., 2024a) and (Chen et al., 2024) from previous studies as training data. The former contains 50,000 samples, while the latter consists of 274,951 samples. Each sample includes an anchor, a positive sample, and a negative sample. Unless otherwise specified, the AutoRegEmbed results presented in the experiment section are based on training with 50,000 samples.

Baselines We categorize the baselines into three groups: (1) models without contrast training, including base models with various embedding methods using the same instructions as AutoRegEmbed and prompt-adjusted embedded models, including Echo (Springer et al., 2024), PromptEOL (Jiang et al., 2024), MetaEOL (Lei et al., 2024), and GenEOL (Thirukovalluru and Dhingra, 2024); (2) unsupervised contrast training models, primarily LLM2Vec (BehnamGhader et al., 2024) with different base models; and (3) supervised contrast training models, which consist of NV-Embed (Lee et al., 2024), SFR-Embedding-2_R (Meng et al., 2024), gte-Qwen2-7B-instruct (Li et al., 2023), LLM2Vec (BehnamGhader et al., 2024), and fair baselines.

4.2 Main Results

Table 1 summarizes the results of various baselines and AutoRegEmbed on ten STS datasets, along with the training data required for each method.

AutoRegEmbed vs. Without Contrastive Training Models without contrastive training are divided into two categories. The first is our own fair baseline model, which performs significantly worse than AutoRegEmbed, with an average performance 20% lower. This highlights the difficulty of untrained LLMs in directly generating high-quality embeddings. While some methods enhance the base model's embeddings through prompt optimization, their improvements remain limited—even on a 13B-parameter model—and come with significant additional reasoning costs. For instance, Echo requires processing text twice to mitigate unidirectional attention limitations, MetaEOL aggregates embeddings from eight different instructions, and GenEOL relies on ChatGPT to generate up to 32 text variants before aggregating their embeddings. These additional reasoning steps severely constrain their practical implementation.

AutoRegEmbed vs. Unsupervised Contrastive Training LLM2Vec enhances existing LLMs using an unsupervised contrastive learning approach similar to SimCSE, leading to significant performance gains. Compared to the base model, the unsupervised version of LLM2Vec improves performance by over 15%. Although it utilizes almost 160,000 data samples, its performance remains 4.5% lower than AutoRegEmbed, demonstrating its lower efficiency.

Method	Params	BIOSSES	SICK-R	STS12	STS13	STS14	STS15	STS16	STS17	STS22	STS-B	Avg.	Vol.
Without Contrastive Training													
LLaMA2-L	7B	63.29	65.10	45.26	70.83	56.69	62.48	63.27	49.76	-7.76	60.43	$60.58_{(7)}/56.91_{(10)}$	0
LLaMA2-M	7B	65.96	60.01	44.76	64.13	48.66	62.33	63.16	64.35	27.59	53.50	$56.65_{(7)}/58.67_{(10)}$	0
Mistral-v0.1-L	7B	54.40	67.40	48.54	64.27	54.89	65.05	62.12	48.22	13.71	63.05	$60.76_{(7)}/56.20_{(10)}$	0
Mistral-v0.1-M	7B	67.46	62.42	50.11	66.45	52.60	61.93	65.02	71.28	29.79	54.19	$58.96_{(7)}/61.13_{(10)}$	0
Echo-LLaMA2	7B	-	64.39	52.40	72.40	61.24	72.67	73.51	-	-	65.73	$66.05_{(7)}$/-	0
Echo-LLaMA2	13B	-	70.27	59.36	79.01	69.75	79.86	76.75	-	-	71.31	$72.33_{(7)}$/-	0
PromptEOL-LLaMA2	7B	-	69.64	58.81	77.01	66.34	73.22	73.56	-	-	71.66	$70.03_{(7)}$/-	0
PromptEOL-Mistral	7B	-	69.47	63.08	78.58	69.40	77.92	79.01	-	-	75.77	$73.32_{(7)}$/-	0
PromptEOL-LLaMA3	8B	-	60.88	68.94	78.57	68.18	76.75	77.16	-	-	72.83	$71.90_{(7)}$/-	0
PromptEOL-LLaMA2	13B	-	68.23	56.19	76.42	65.42	72.73	75.21	-	-	67.96	$68.83_{(7)}$/-	0
MetaEOL-LLaMA2	7B	-	74.86	64.16	81.61	73.09	81.11	78.94	-	-	77.96	$75.96_{(7)}$/-	0
MetaEOL-Mistral	7B	-	75.13	64.05	82.35	71.57	81.36	79.85	-	-	78.29	$76.09_{(7)}$/-	0
GenEOL-LLaMA2-Mistral	7B	-	78.08	70.24	83.43	78.03	81.79	80.65	-	-	80.46	$78.95_{(7)}$/-	0
GenEOL-LLaMA2-ChatGPT	7B	-	78.71	70.78	83.28	77.75	82.10	80.45	-	-	79.83	$78.99_{(7)}$/-	0
Unsupervised Contrastive Training													
LLM2Vec-LLaMA2♣	7B	82.41	71.77	65.39	79.26	72.98	82.72	81.02	86.70	63.47	78.32	$75.92_{(7)}/76.41_{(10)}$	~160,000
LLM2Vec-Mistral♣	7B	83.29	75.55	67.65	83.90	76.97	83.80	81.91	85.58	65.93	80.42	$78.60_{(7)}/78.50_{(10)}$	~160,000
Supervised Contrastive Training													
NV-Embed♣	7.73B	85.59	82.80	76.22	86.30	82.09	87.24	84.77	87.42	69.85	86.14	$83.65_{(7)}/82.84_{(10)}$	1,054,000
SFR-Embedding-2_R♣	7B	87.60	77.01	75.67	82.40	79.93	85.82	84.50	88.93	67.10	83.60	$81.28_{(7)}/81.26_{(10)}$	~1,751,000
gte-Qwen2-7B-instruct♣	7.49B	81.37	79.16	79.53	88.97	83.87	88.48	86.49	88.75	67.16	86.81	$84.76_{(7)}/83.06_{(10)}$	~791,000,000
LLM2Vec-LLaMA2♣	7B	82.13	83.01	78.85	86.84	84.04	88.72	86.79	90.63	67.55	88.72	$85.28_{(7)}/83.73_{(10)}$	544,000
LLM2Vec-Mistral♣	7B	85.24	83.70	78.80	86.37	84.04	88.99	87.22	90.19	67.68	88.65	$85.40_{(7)}/84.01_{(10)}$	544,000
LLaMA2-L	7B	77.58	77.85	73.72	84.04	79.82	85.03	84.78	87.53	26.87	86.18	$81.63_{(7)}/76.34_{(10)}$	50,000
LLaMA2-inbatch-L	7B	78.81	82.76	77.70	85.01	81.82	88.30	86.12	90.53	20.70	87.94	$84.24_{(7)}/77.97_{(10)}$	50,000
LLaMA2-M	7B	75.65	78.92	74.12	84.17	80.00	85.63	83.28	85.65	65.09	86.27	$81.77_{(7)}/79.88_{(10)}$	50,000
LLaMA2-inbatch-M	7B	78.09	83.17	77.10	82.82	80.53	87.40	84.43	90.02	64.59	87.18	$83.23_{(7)}/81.53_{(10)}$	50,000
LLaMA2-inbatch-M	7B	77.43	82.26	77.95	84.90	82.06	87.22	86.43	88.22	66.42	86.12	$83.85_{(7)}/81.90_{(10)}$	274,951
Information Compression and Conditional Distribution Alignment													
AutoRegEmbed-LLaMA2	7B	85.50	79.07	79.57	86.90	83.28	88.45	86.57	88.61	66.16	86.59	$84.35_{(7)}/83.07_{(10)}$	50,000(16,382)
AutoRegEmbed-Mistral	7B	86.69	80.21	78.33	86.22	82.36	88.42	86.43	88.70	64.27	87.05	$84.15_{(7)}/82.87_{(10)}$	50,000(16,382)
AutoRegEmbed-LLaMA2	7B	85.62	81.93	78.84	86.76	84.01	89.43	87.72	89.04	66.77	87.96	$85.24_{(7)}/83.81_{(10)}$	274,951(16,382)

Table 1: Results on STS tasks (Spearman correlation scaled by 100x). The parentheses in the **Avg.** column indicate the number of datasets used to compute the average. **Vol.** denotes the number of training triplets, while the numbers in brackets indicate the instruction fine-tuning data used by AutoRegEmbed during the information compression stage. The symbol "~" denotes an estimated value. "" represents our own fair baselines, and we apply a grid search to ensure optimal performance. "-L" and "-M" denote the hidden state of the last token and the average pooling of all token hidden states, respectively. The symbol ♣ indicates that not all data are open source.

AutoRegEmbed vs. Supervised Contrastive Training Supervised contrastive learning is the mainstream approach for building high-quality embedding models. We first compared top-performing methods that once ranked on the MTEB leaderbord (with the time they reached SOTA in brackets), including NV-Embed (2024.08), SFR-Embedding-2_R (2024.02), gte-Qwen2-7B-instruct (2024.06), and LLM2Vec (2024.05). In terms of model performance, AutoRegEmbed outperforms most SOTA methods on ten STS datasets, trailing only 0.2% behind the best version of LLM2Vec. From a data efficiency perspective, AutoRegEmbed achieves performance comparable to the previous SOTA models with just 66,382 training samples, whereas the latter requires tens of millions of triplets to reach peak performance. Additionally, previous SOTA models employ multi-task learning (e.g., retrieval and clustering), whose impact on STS performance remains unclear. To ensure a fair comparison, we use single-task contrastive learning as a baseline. Unlike traditional contrastive learning, AutoRegEmbed does not rely on in-batch negative samples. So we add two baselines to single-task contrastive learning that also exclude the in-batch negative sample strategy. As shown in Table 1, even under identical training data, AutoRegEmbed outperforms four different single-task contrastive learning, further validating its effectiveness.

4.3 Ablation Study

To verify the effectiveness of AutoRegEmbed, we conducted an ablation study. First, we removed Conditional Distribution Alignment to evaluate its impact on model performance. Second, since Equation 2 was derived empirically in our previous work, we tested different variants of this equation to confirm that it remains the optimal choice. Different variants of Equation 2 include **Log_sigmoid**, which maps similarity to a logarithmic scale for integration with the exponential function e, as well as **KL divergence** and **JS divergence**, which quantify the distance between the conditional probability

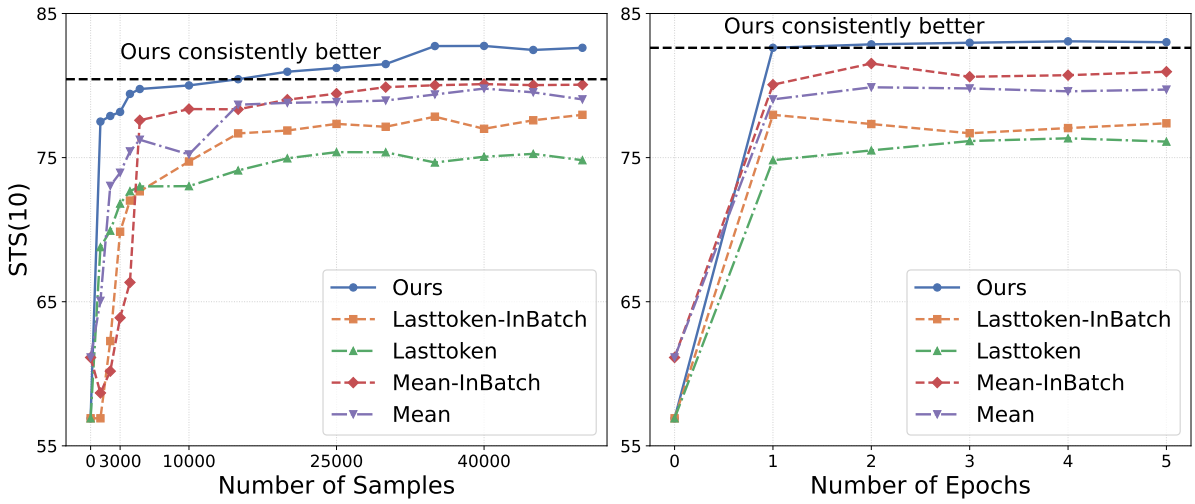

Figure 3: We evaluate the learning efficiency of our method against traditional contrastive learning on 10 STS datasets, comparing their performance under the same number of samples (left figure) and same number of epochs (right figure). Further details are provided in Appendix A.

Method	**Avg.**
AutoRegEmbed-LLaMA2	$83.07_{(10)}$
Tasks	
w/o Conditional Distribution Alignment	$73.90_{(10)}$
LLaMA2-L (Without Training)	$56.91_{(10)}$
Equation 2	
Log_sigmoid	$82.93_{(10)}$
KL divergence	$79.82_{(10)}$
JS divergence	$79.02_{(10)}$

Table 2: Ablation experiments of AutoRegEmbed. We conduct ablation and contrast experiments on various tasks and Equation 2 to demonstrate the effectiveness of AutoRegEmbed.

distributions of positive and negative sample embeddings in distinct ways. The specific equations are provided in Appendix B. Table 2 presents the ablation results. The experiments on different tasks indicate that Conditional Distribution Alignment improves performance by 9.17%, while Information Compression contributes a 16.99% improvement, demonstrating the effectiveness of both tasks. Additionally, experiments on variants of Equation 2 reveal that, although using a logarithmic scale for similarity and employing KL or JS divergence to measure distribution distance are more intuitive approaches, they do not surpass the performance of the original loss function in Equation 2. Thus, Equation 2 can be regarded as a more effective loss function.

4.4 Learning Efficiency

To verify that AutoRegEmbed is better suited for LLMs, we compare its performance with four contrastive learning baselines under the same training data and the same number of epochs, as shown in Figure 3. The left figure in Figure 3 shows that as the training data increases, the performance of both AutoRegEmbed and other contrastive learning methods improves, but AutoRegEmbed exhibits the fastest growth. Notably, with just 15,000 samples, AutoRegEmbed already surpasses the maximum performance of other contrastive learning models. The right figure demonstrates that as the number of epochs increases, AutoRegEmbed also improves at the fastest rate. These results indicate that AutoRegEmbed significantly outperforms the baseline models in learning efficiency.

5 Conclusions

To address the limitation that traditional contrastive learning does not adhere to the autoregressive nature of LLMs, we propose AutoRegEmbed—a novel contrastive learning method based on embedded conditional probability distributions. AutoRegEmbed ensures that LLM-generated embeddings capture global semantics while maintaining alignment and uniformity through information compression and conditional distribution alignment tasks. AutoRegEmbed achieves comparable performance to SOTA models with fewer training samples and superior learning efficiency.

6 Limitations

The primary advantage of AutoRegEmbed lies in its ability to effectively harness the power of large language models (LLMs) to construct robust and high-quality text embeddings. However, it is important to acknowledge several limitations of our approach.

AutoRegEmbed does not possess inherent mechanisms to filter or detect malicious or harmful content in the data it processes. While the model is capable of generating embeddings from a wide range of text inputs, it lacks the ability to evaluate the ethical or safety implications of the data. This makes it vulnerable to issues related to biased, offensive, or otherwise problematic content present in the training corpus. In cases where the training data contains harmful or discriminatory material, the embeddings generated by AutoRegEmbed may inadvertently carry forward these biases, potentially leading to unintended and undesirable outcomes when applied to real-world tasks.

To mitigate this risk, we recommend that users of AutoRegEmbed ensure that the training data is carefully curated, and ideally, filtered for harmful content. By using safe and ethically sourced data, the model's potential for propagating bias or harm can be minimized. Additionally, users should be cautious when applying AutoRegEmbed to sensitive domains, where the generation of unsafe or biased embeddings could have significant consequences.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2015, Denver, Colorado, USA, June 4-5, 2015*, pages 252–263. The Association for Computer Linguistics.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *Proceedings of the 8th International Workshop on Semantic Evaluation, SemEval@COLING 2014, Dublin, Ireland, August 23-24, 2014*, pages 81–91. The Association for Computer Linguistics.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2016, San Diego, CA, USA, June 16-17, 2016*, pages 497–511. The Association for Computer Linguistics.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *Proceedings of the 6th International Workshop on Semantic Evaluation, SemEval@NAACL-HLT 2012, Montréal, Canada, June 7-8, 2012*, pages 385–393. The Association for Computer Linguistics.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Proceedings of the Second Joint Conference on Lexical and Computational Semantics, *SEM 2013, June 13-14, 2013, Atlanta, Georgia, USA*, pages 32–43. Association for Computational Linguistics.

Akari Asai, Timo Schick, Patrick S. H. Lewis, Xilun Chen, Gautier Izacard, Sebastian Riedel, Hannaneh Hajishirzi, and Wen-tau Yih. 2023. Task-aware retrieval with instructions. In *Findings of the Association for Computational Linguistics: ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 3650–3675. Association for Computational Linguistics.

Parishad BehnamGhader, Vaibhav Adlakha, Marius Mosbach, Dzmitry Bahdanau, Nicolas Chapados, and Siva Reddy. 2024. Llm2vec: Large language models are secretly powerful text encoders. *CoRR*, abs/2404.05961.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *Proceedings of the 11th International Workshop on Semantic Evaluation, SemEval@ACL 2017, Vancouver, Canada, August 3-4, 2017*, pages 1–14. Association for Computational Linguistics.

Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *CoRR*, abs/2402.03216.

Xi Chen, Ali Zeynali, Chico Q. Camargo, Fabian Flöck, Devin Gaffney, Przemyslaw A. Grabowicz, Scott Hale, David Jurgens, and Mattia Samory. 2022. Semeval-2022 task 8: Multilingual news article similarity. In *Proceedings of the 16th International Workshop on Semantic Evaluation, SemEval@NAACL 2022, Seattle, Washington, United States, July 14-15, 2022*, pages 1094–1106. Association for Computational Linguistics.

Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting language models to compress contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 3829–3846. Association for Computational Linguistics.

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Jingcheng Deng, Liang Pang, Huawei Shen, and Xueqi Cheng. 2023. Regavae: A retrieval-augmented gaussian mixture variational auto-encoder for language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 2500–2510. Association for Computational Linguistics.

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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Hanxing Ding, Liang Pang, Zihao Wei, Huawei Shen, and Xueqi Cheng. 2024. Retrieve only when it needs: Adaptive retrieval augmentation for hallucination mitigation in large language models. *CoRR*, abs/2402.10612.

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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2024. In-context autoencoder for context compression in a large language model. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. Deberta: decoding-enhanced bert with disentangled attention. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2024. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, pages 3182–3196. Association for Computational Linguistics.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. 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 2020, Online, November 16-20, 2020*, pages 6769–6781. Association for Computational Linguistics.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *CoRR*, abs/2405.17428.

Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. 2024. Meta-task prompting elicits embeddings from large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 10141–10157. Association for Computational Linguistics.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 9119–9130. Association for Computational Linguistics.

Chaofan Li, Zheng Liu, Shitao Xiao, Yingxia Shao, and Defu Lian. 2024a. Llama2vec: Unsupervised adaptation of large language models for dense retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 3490–3500. Association for Computational Linguistics.

Chaofan Li, Minghao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. 2024b. Making text embedders few-shot learners. *CoRR*, abs/2409.15700.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards

general text embeddings with multi-stage contrastive learning. *CoRR*, abs/2308.03281.

Tian Yu Liu, Matthew Trager, Alessandro Achille, Pramuditha Perera, Luca Zancato, and Stefano Soatto. 2024. Meaning representations from trajectories in autoregressive models. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Xueguang Ma, Liang Wang, Nan Yang, Furu Wei, and Jimmy Lin. 2024. Fine-tuning llama for multi-stage text retrieval. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pages 2421–2425. ACM.

Rui Meng, Ye Liu, Shafiq Rayhan Joty, Caiming Xiong, Yingbo Zhou, and Semih Yavuz. 2024. Sfr-embedding-2: Advanced text embedding with multi-stage training.

Jesse Mu, Xiang Li, and Noah D. Goodman. 2023. Learning to compress prompts with gist tokens. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Niklas Muennighoff, Hongjin Su, Liang Wang, Nan Yang, Furu Wei, Tao Yu, Amanpreet Singh, and Douwe Kiela. 2024. Generative representational instruction tuning. *CoRR*, abs/2402.09906.

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, EACL 2023, Dubrovnik, Croatia, May 2-6, 2023*, pages 2006–2029. Association for Computational Linguistics.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y. Zhao, Yi Luan, Keith B. Hall, Ming-Wei Chang, and Yinfei Yang. 2022. Large dual encoders are generalizable retrievers. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 9844–9855. Association for Computational Linguistics.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

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 2019, Hong Kong, China, November 3-7, 2019*, pages 3980–3990. Association for Computational Linguistics.

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), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 8371–8384. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition improves language model embeddings. *CoRR*, abs/2402.15449.

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, Toronto, Canada, July 9-14, 2023*, pages 1102–1121. Association for Computational Linguistics.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *CoRR*, abs/2103.15316.

Raghuveer Thirukovalluru and Bhuwan Dhingra. 2024. Geneol: Harnessing the generative power of llms for training-free sentence embeddings. *CoRR*, abs/2410.14635.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

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. 2024a. Improving text embeddings with large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2024, Bangkok, Thailand, August 11-16, 2024*, pages 11897–11916. Association for Computational Linguistics.

Tongzhou Wang and Phillip Isola. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 9929–9939. PMLR.

Zhichao Wang, Bin Bi, Shiva Kumar Pentyala, Kiran Ramnath, Sougata Chaudhuri, Shubham Mehrotra, Zixu James Zhu, Xiang-Bo Mao, Sitaram Asur, and Na Claire Cheng. 2024b. A comprehensive survey of LLM alignment techniques: Rlhf, rlaif, ppo, DPO and more. *CoRR*, abs/2407.16216.

Shengyao Zhuang, Xueguang Ma, Bevan Koopman, Jimmy Lin, and Guido Zuccon. 2024. Promptreps: Prompting large language models to generate dense and sparse representations for zero-shot document retrieval. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, EMNLP 2024, Miami, FL, USA, November 12-16, 2024*, pages 4375–4391. Association for Computational Linguistics.

A Implementation details

AutoRegEmbed For the information compression task, we set the learning rate to 2e-5, the batch size to 32, and train for 2 epoch. To represent the semantics of the input, we use 5 compressed tokens. For the conditional distribution alignment task, the learning rate is set to 5e-6, with a batch size of 32 and 4 epochs. The temperature parameters t and b are both set to 0.1. For the above two tasks, we set the maximum token length of context, instruction, and target to 512. Furthermore, we employ the bfloat16 format, enable FlashAttention 2, and train on four A100-80G GPUs with DeepSpeed and Zero-2. The information compression task takes 20 minutes, while the conditional distribution alignment task, involving 50,000 samples, takes approximately 1 hour.

Fair Comparative Learning Baselines We train our own fair contrastive learning baseline based on the standard InfoNCE loss, with some code available in the FlagEmbedding repository[1]). For baselines utilizing the in-batch negative sample strategy (LLaMA2-inbatch-L and LLaMA2-inbatch-M), we experimented with batch sizes of 128, 256, 512, and 1024, determining that 512 yields the best performance. Additionally, we ensure that gradients are propagated across different devices. For baselines that do not use the in-batch negative sample strategy, we set the batch size to 32, maintaining consistency with AutoRegEmbed. Regarding the learning rate, we tested 1e-5, 5e-5, 1e-4, and 2e-4, finding that 1e-4 delivers the best results. All training data is consistent with AutoRegEmbed. We train the fair contrastive learning baseline using DeepSpeed and Zero-2 on four A100-80G GPUs in 1 hour.

[1] https://github.com/FlagOpen/FlagEmbedding

B Variants of Equation 2

This section explores various possible modifications and extensions of Equation 2.

Log_sigmoid Given that most loss functions are logarithmic in nature, we can modify the similarity function in Equation 2 by replacing the sigmoid with a Log-Sigmoid function, resulting in a more interpretable formulation:

$$\mathcal{L}_{\text{CDA}} = \mathbb{E}[-\log \frac{e^{S_1(q,d^+)/\tau}}{e^{S_1(q,d^+)/\tau} + \sum_i e^{S_2(d^+,d_i^-;q)/\tau}}],$$

$$S_1(q,d^+) = -\log\sigma(\beta\, |\log \frac{p_{\theta_E}(d^+|e_{q,I_{\text{next}}})}{p_{\theta_E}(d^+|e_{d^+,I_{\text{self}}})}|),$$

$$S_2(d^+,d_i^-;q) = -\log\sigma(\beta \log \frac{p_{\theta_E}(d^+|e_{q,I_{\text{next}}})}{p_{\text{ref}}(d^+|e_{q,I_{\text{next}}})} - \beta \log \frac{p_{\theta_E}(d^-|e_{q,I_{\text{next}}})}{p_{\text{ref}}(d^-|e_{q,I_{\text{next}}})}).$$

KL divergence We also experimented with replacing the difference in log probabilities with the KL divergence between the conditional probability distributions:

$$\mathcal{L}_{\text{CDA}} = \mathbb{E}[-\log \frac{e^{S_1(q,d^+)/\tau}}{e^{S_1(q,d^+)/\tau} + \sum_i e^{S_2(q,d_i^-)/\tau}}],$$

$$S_1(q,d^+) = -\sigma(\frac{1}{T}\sum_{t=1}^{T} \text{KL}(p_{\theta_E}(d_t^+|d_{<t}^+, e_{d^+,I_{\text{self}}}), p_{\theta_E}(d_t^+|d_{<t}^+, e_{q,I_{\text{next}}}))),$$

$$S_2(q,d^-) = -\sigma(\frac{1}{T}\sum_{t=1}^{T} \text{KL}(p_{\theta_E}(d_t^+|d_{<t}^-, e_{d^-,I_{\text{self}}}), p_{\theta_E}(d_t^+|d_{<t}^+, e_{q,I_{\text{next}}}))).$$

JS divergence In addition to KL divergence, we also employed JS divergence as a measure of dis-

tribution distance:

$$\mathcal{L}_{\text{CDA}} = \mathbb{E}[-\log \frac{e^{S_1(q,d^+)/\tau}}{e^{S_1(q,d^+)/\tau} + \sum_i e^{S_2(q,d_i^-)/\tau}}],$$

$$S_1(q, d^+) = -\sigma(\frac{1}{T}\sum_{t=1}^{T} \text{JS}(p_{\theta_E}(d_t^+ | d_{<t}^+, e_{d^+,I_{\text{self}}}),$$

$$p_{\theta_E}(d_t^+ | d_{<t}^+, e_{q,I_{\text{next}}}))),$$

$$S_2(q, d^-) = -\sigma(\frac{1}{T}\sum_{t=1}^{T} \text{JS}(p_{\theta_E}(d_t^+ | d_{<t}^-, e_{d^-,I_{\text{self}}}),$$

$$p_{\theta_E}(d_t^+ | d_{<t}^+, e_{q,I_{\text{next}}}))).$$