A Text is Worth Several Tokens: Text Embedding from LLMs Secretly Aligns Well with The Key Tokens

Zhijie Nie[1,3], **Richong Zhang**[1,2*], **Zhanyu Wu**[1]
[1]CCSE, School of Computer Science and Engineering, Beihang University, Beijing, China
[2]Zhongguancun Laboratory, Beijing, China
[3]Shen Yuan Honors College, Beihang University, Beijing, China
{niezj,zhangrc,wuzy24}@act.buaa.edu.cn

Abstract

Text embeddings from large language models (LLMs) have achieved excellent results in tasks such as information retrieval, semantic textual similarity, etc. In this work, we show an interesting finding: when feeding a text into the LLM-based embedder, the obtained text embedding can be aligned with the key tokens in the input text. We first fully analyze this phenomenon on eight LLM-based embedders and show that this phenomenon is universal and is not affected by model architecture, training strategy, and embedding method. Upon further analysis, we find that the main change in embedding space between these embedders and their LLM backbones lies in the first principal component. By adjusting the first principal component, we can align text embedding with the key tokens. Finally, we demonstrate the broad application potential of this finding: (1) we propose a simple and practical sparse retrieval method based on the aligned tokens, which can achieve 80% of the dense retrieval effect of the same model while reducing the computation significantly; (2) we show that our findings provide a novel perspective to help understand novel technologies (e.g., instruction-following embedding) and fuzzy concepts (e.g., semantic relatedness vs. similarity) in this field[1].

1 Introduction

Large language models (LLMs) have recently made rapid progress on various natural language understanding tasks using the generative paradigm (Brown et al., 2020). However, not all tasks lend themselves to the generative paradigm in practice; tasks such as information retrieval, text clustering, and semantic text similarity usually rely on high-quality text embeddings. Thus, more and more attention has been focused on obtaining high-quality textual embeddings from large language models (Jiang et al., 2023; Springer et al., 2024; BehnamGhader et al., 2024).

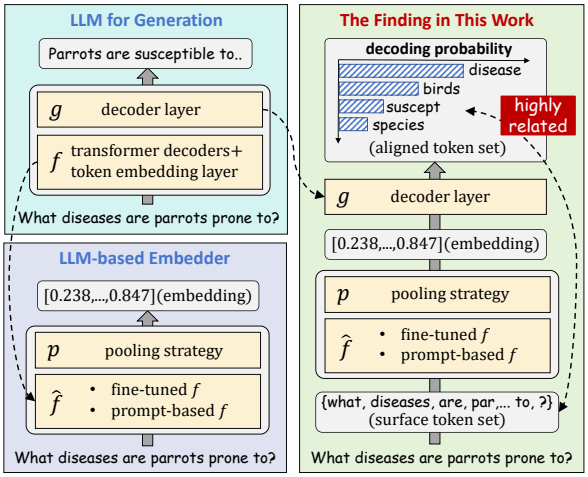


Figure 1: Paradigms on LLMs for text generation and embedding (left) and our novel findings (right).

As shown on the left half of Figure 1, the LLM for generation takes the texts as input and output. The input text is tokenized and passed through the module f to obtain its hidden states. Then, a decoder layer g is required, which maps the high-dimensional hidden states to the vocabulary-length logits and computes the decoded probability for each token. When LLMs are converted for text embedding, current methods typically incorporate the following changes: (1) g is discarded because there is no need to map to the vocabulary; (2) f is converted into $\hat{f}$ using prompt-engineering (Jiang et al., 2023; Springer et al., 2024) or contrastive learning (Muennighoff, 2022; BehnamGhader et al., 2024); and (3) a pooling strategy p is used to weight sum of hidden states and obtain the text embedding.

In this paper, we are not proposing a new text embedding method for LLMs. Instead, our research centers on a very interesting finding: when the text embedding obtained by $\hat{f}$ passes through the de-

* Corresponding author

[1]Our code is available at https://github.com/Arthurizijar/Text_aligns_tokens

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 7683–7694
July 27 - August 1, 2025

coder layer g from the same LLM, the tokens with the highest decoding probability are highly related to the input text. In other words, the embedding of the input text is aligned with some key tokens of that text. As shown in the right half of Figure 1, when the input text is "*What diseases are parrots prone to ?*", we can find the literally-related tokens, such as "*disease*" and the semantically-related tokens, such as "*birds*" and "*suscept*" have the highest decoding probabilities.

This phenomenon may not be surprising in some prompt-based methods, which direct LLMs to summarise the whole text in a word (See §2.2 for details). However, based on the sufficient study of eight LLM-based embedders [2], we observe that the above phenomenon is universal, independent of the LLMs' architecture, the training strategy, and the embedding method. (§3). Especially this phenomenon appears even more clearly in those methods based on contrastive learning, uncovering the unity among different methods.

Considering the unusual consistency of this phenomenon, we perform deeper analyses based on these LLMs to understand this finding more precisely. Specifically, we compare the embedding spaces of f and $\hat{f}$ using spectral analysis (§4). We find that the dominant change in $\hat{f}$ is concentrated in the first principal component. By manually adjusting the first principal component of the embedding space, we can replicate the phenomenon of aligning text embeddings to key tokens.

With a deeper understanding of our findings, we believe that it has a rich potential for application (§5). For example, we find that the criticism of LLM-generated embedding mainly stems from its high dimensionality, resulting in significant inference and storage overhead (Muennighoff et al., 2024). To address this, we propose a new sparse retrieval method based on our findings. We convert document embeddings into a sparse representation consisting only of aligned tokens and utilize a few aligned tokens from the query embedding for expansion. Despite its simplicity, our method achieves over 80% of the performance of the original LLM-based embedder. At the same time, we show that our work helps to intuitively understand (1) the working mechanism of the instruction-following embedding (Su et al., 2023) and (2) the influence of training data on the embedding space.

[2]We use "embedder" instead of "encoder" to prevent unnecessary misunderstanding since the backbones of the current methods are usually decoder-only LLMs.

Our contributions are summarized as follows:

- We find that the text embeddings obtained in the LLM-based embedders align with the key tokens, providing a unified perspective for understanding prompt engineering methods and contrastive learning methods;
- We explain why this phenomenon occurs from the perspective of spectral analysis and find that the current method mainly changes the first principal component of the original embedding space of the LLMs;
- We show a series of application examples, including improvements to the method and interpretability of the model, demonstrating the large application potential of our findings.

2 Background

2.1 Basic Paradigm

Given a LLM F, we can divide it into two parts:

$$F = g \circ f \tag{1}$$

where g is the decoder layer, and f is the rest modules of the LLM. In the existing LLM embedding methods, g is discarded, while f can be used as a text embedder. Given a text s_i, we convert it to a token sequence using LLM's tokenizer and get $s_i = \{t_{i1}, \cdots, t_{il}\}$, where l is the sequence length; then we can get the hidden state of the last layer:

$$\mathbf{H} = [\mathbf{h}_{i1}^{(t)}, \cdots, \mathbf{h}_{il}^{(t)}] = f(s_i) \tag{2}$$

where $\mathbf{H} \in \mathbb{R}^{d \times l}$ and $\mathbf{h}_{ij}^{(t)} \in \mathbb{R}^{d \times 1}$ is the i-th d-dimensional hidden state. Subsequently, the pooling strategy $p(.)$ is used to $\mathbf{H}$ for the text embedding $\mathbf{h}_i$, which can be expressed as

$$\mathbf{h}_i = p(f(s_i)) = p(\mathbf{H}) = \sum\nolimits_{j=1}^{l} \alpha_j \mathbf{h}_{ij}^{(t)} \tag{3}$$

where $\{\alpha_j\}_{j=1}^{l}$ is the weight factor satisfying $\sum_{j=1}^{l} \alpha_j = 1$. Specifically, there are three popular pooling strategies in practice: for last pooling, α_j is 1 if $j = l$ else is 0; for mean pooling, $\alpha_j = 1/l$ for each j; for weighted mean pooling (Muennighoff, 2022), $\alpha_j = j / \sum_{j=1}^{l} j$.

However, text embeddings obtained directly from the encoder f show poor performance. It is unsurprising since the pre-training task, next token prediction, is not designed for embedding, and the unidirectional attention detracts from the

expressive power of the hidden states (Li and Li, 2024). In the subsequent subsections, we introduce how the existing methods improve the embedding's quality based on the top of f. For simplicity, we indiscriminately refer to the LLM-based embedder improved based on f as $\hat{f}$.

2.2 Embedding via Prompt Engineering

The embedder $\hat{f}$ based on prompt engineering fills the text into prompt templates to improve the quality of text embedding, which can be expressed as

$$\hat{f}(s_i) = f(t(s_i)) \tag{4}$$

where $t(.)$ represents the operation of filling the text into a fixed prompt template.

PromptEOL (Jiang et al., 2023) introduces a prompt template: `This sentence:"[text]" means in one word:"`, where `[text]` is a placeholder. In practice, the template where `[text]` is replaced by a specific text is sent into the encoder f, and the last pooling strategy is used to obtain the text embedding. The following works design a better prompt template based on task-oriented (Lei et al., 2024) or chain-of-thought (Zhang et al., 2024) can lead to better performance.

The methods based on prompt engineering are simple and training-free, so they are unlikely to compromise the LLMs' generation capabilities. However, they provide limited performance improvement for downstream tasks.

2.3 Embedding via Contrastive Learning

The methods based on contrastive learning inherited the valuable experience of the BERT-based encoder era (Gao et al., 2021). In these methods, $\hat{f}$ is fine-tuned f with contrastive learning. Due to the large parameter count of f itself, parameter-efficient fine-tuning methods such as LoRA (Hu et al., 2021) are usually used.

Given a text dataset D, for any text $s_i \in D$, we first obtain its embedding $\mathbf{h}_i$ from f with a specific pooling strategy. Then positive pairs $(\mathbf{h}_i, \mathbf{h}_i^+)$ and negative pairs $\{(\mathbf{h}_i, \mathbf{h}_{ij}^-)\}_{j=1}^N$ are constructed following different settings, where N is the negative example number. In the unsupervised setting, two data-augmented views of a text are considered a positive pair, while the negative samples are randomly sampled from the datasets. In the supervised setting, the positive pair is a labeled text pair, which can be query-document, question-answer or hypothesis-entailment, while hard negative pairs may be introduced. Finally, the contrastive loss can be expressed as

$$\mathcal{L}_{\text{cl}} = -\log \frac{e^{d(\mathbf{h}_i, \mathbf{h}_i^+)/\tau}}{e^{d(\mathbf{h}_i, \mathbf{h}_i^+)/\tau} + \sum_{j=1}^N e^{d(\mathbf{h}_i, \mathbf{h}_{ij}^-)/\tau}} \tag{5}$$

where $d(\cdot, \cdot)$ is a distance function, τ is the temperature hyper-parameter. During fine-tuning, the contrastive loss draws positive text pairs close while pushing negative text pairs away.

Additional Tricks There are some effective tricks in the existing works, which include: (1) switching casual attention to bi-directional attention (BehnamGhader et al., 2024); (2) using different instruction prefixes for the datasets from different tasks to minimize inter-task interference (Su et al., 2023); (3) co-training contrastive learning and next word prediction to minimize reductions to generative capability (Muennighoff et al., 2024).

3 Embedding Aligns with Key Tokens

3.1 Motivation

To analyze the pre-trained transformer in the embedding space, Elhage et al. (2021); Geva et al. (2022); Dar et al. (2022) attempt to multiply the attention or feed-forward layer parameters with the token embedding matrix to explain how these parameters work. For example, Geva et al. (2022) multiplies the feed-forward value vector with the token embedding matrix to obtain a distribution over the vocabulary and find that the tokens with high probability can explain what FFNs update to hidden layer representations. Inspired by these works, we try to interpret text embeddings obtained from LLMs by mapping them into the token space.

3.2 Method

To implement the above idea, we introduce a text dataset D, and a triplet $(\hat{f}, T, \mathbf{E}_g)$: $\hat{f}$ is the LLM-based embedder, $T = \{t_1, \cdots, t_L\}$ is the L-sized vocabulary and $\mathbf{E}_g = [\mathbf{e}_{t_1}, \cdots, \mathbf{e}_{t_L}] \in \mathbb{R}^{d \times L}$ is the token embedding matrix from the decoded layer g, where $\mathbf{e}_{t_j} \in \mathbb{R}^{d \times 1}$ is the token embedding of token t_j. Note that T and $\mathbf{E}_g$ are determined by the original LLM F and $\mathbf{E}_g$ is the only parameter in g [3], therefore, there is no difference between $\mathbf{E}_g^\top \mathbf{h}_i$ and $g(\mathbf{h}_i)$ for any text embedding $\mathbf{h}_i \in \mathbb{R}^{d \times 1}$.

[3] To the best of our knowledge, all popular LLMs follow the original design of the decoder layer from GPT (Radford et al., 2018), i.e., a linear layer without bias, which also can be regarded as a token embedding matrix.

Model	Architecture		Fine-tuning		Embedding	
	Backbone	Attention	Paradigm	Corpus	Pooling	Similarity
SGPT_{nli}	GPT-Neo	casual	SCL	NLI	weighted mean	cosine
$\text{SGPT}_{\text{msmarco}}$	(1.3B)	casual	SCL	MS MARCO	weighted mean	cosine
OPT_{EOL}	OPT	casual	PE	-	last token	dot product
$\text{OPT}_{\text{EOL+CSE}}$	(1.3B)	casual	PE+SCL	NLI	last	dot product
$\text{LLaMA}_{\text{EOL}}$	LLaMA	casual	PE	-	last token	dot product
$\text{LLaMA}_{\text{EOL+CSE}}$	(7B)	casual	PE+SCL	NLI	last	dot product
GritLM	Mistral	bi-directional	SCL+NTP	Tulu 2+E5+S2ORC	mean	cosine
LLM2Vec	(7B)	bi-directional	MNTP→SCL	E5	weighted mean	cosine

Table 1: Detailed information on the model used to study the embedding space. The paradigms are shortened as follows: supervised contrastive learning (SCL), unsupervised contrastive learning (UCL), prompt engineering (PE), next token prediction (NTP), and masked next token prediction (MNTP) (BehnamGhader et al., 2024) separately.

Given a text $s_i \in D$, we obtain its literal token set T_{s_i} and top K aligned token set $\hat{T}^K_{s_i}$ then capture the potential connection between these two sets. For T_{s_i}, we (1) convert s_i into tokens by the tokenizer of f and (2) deduplicate the token sequence to form a token set T_{s_i}. For $\hat{T}^K_{s_i}$, we (1) follow the pooling strategy of $\hat{f}$ to obtain the text embedding $\mathbf{h}_i$, (2) calculate the dot product between $\mathbf{h}_i$ and the token embedding $\mathbf{e}_{t_j}$ for each token t_j, (3) obtain the ordered token set $\hat{T}_{s_i}$ by sorting in descending order according to dot-product results, and (4) select the first K elements from $\hat{T}_{s_i}$ to form $\hat{T}^K_{s_i}$. We provide an algorithmic form to describe this process precisely:

Algorithm 1 Embedding-Token Alignment Analysis

Input: A text dataset D and the triplet $(\hat{f}, T, \mathbf{E}_g)$.
Initialization: $i \leftarrow 0$, $j \leftarrow 0$
while $i \leq |D|$ **do**
  Get the i-th text s_i in D
  Deduplicate tokenizer(s_i) to obtain T_{s_i}
  Calculate $\mathbf{h}_i \leftarrow \text{pooling}(\hat{f}(s_i))$
  while $j \leq |T|$ **do**
    Calculate $\text{score}(t_j, s_i) \leftarrow \mathbf{e}_{t_j}^\top \mathbf{h}_i$
    Update $j \leftarrow j + 1$
  end while
  Sort T in descending by $\text{score}(t_j, s_i)$ to get $\hat{T}_{s_i}$
  Select the first K elements from $\hat{T}_{s_i}$ to form $\hat{T}^K_{s_i}$
  Update $i \leftarrow i + 1$
end while
Output: T_{s_i} and $\hat{T}^K_{s_i}$

3.3 Experiment

Dataset D We randomly sample 10K of the 1M Wikipedia texts provided by Gao et al. (2021) and report the metric calculated by this dataset. Experiments on other datasets, such as SNLI (Bowman et al., 2015) and MSMARCO (Nguyen et al., 2016), lead to similar conclusions.

Triplet $(\hat{f}, T, \mathbf{E}_g)$ We select eight LLM-based embedders for analysis, which include SGPT_{nli} and $\text{SGPT}_{\text{msmarco}}$ (Muennighoff, 2022); OPT_{EOL}, $\text{OPT}_{\text{EOL+CSE}}$, $\text{LLaMA}_{\text{EOL}}$ and $\text{LLaMA}_{\text{EOL+CSE}}$ (Jiang et al., 2023); GritLM (Muennighoff et al., 2024) and LLM2Vec (BehnamGhader et al., 2024). The key information overview of these models is placed in Table 1. We consider these embedders as $\hat{f}$ and obtain T and $\mathbf{E}_g$ from their LLM backbone. To ensure the generalizability of subsequent conclusions, the embedders selected have different architectures, fine-tuning methods, and embedding methods [4]. Note that none of these embedders goes beyond what we describe in §2.

3.4 Analysis of Aligned Tokens

Qualitative Study We sample an input text from D and show the top 10 aligned tokens of the text embedding, i.e., $\hat{T}^{10}_{s_i}$, in Table 2. We also show the aligned tokens for the original f, using the same pooling strategy as the corresponding $\hat{f}$ for fair comparison. To indicate the relationship between each token and the surface token set T_{s_i}, we use different colors to mark: Green represents that the token is in T_{s_i}; Yellow represents that the token and a token in T_{s_i} are same after stemming or lemmatization [5]; Red represents that the token and all tokens in T_{s_i} have no literal connection. As shown in Table 2, we find that (1) the text embeddings from the original f align with some tokens related T_{s_i}, but most of them are meaningless to-

[4] Regardless of what the similarity metric is recommended, we use a simple matrix multiplication between $\mathbf{E}_g$ and $\mathbf{h}_i$, to ensure consistency with the original decoding process.

[5] We use the tools provided by NLTK (Loper and Bird, 2002): SnowballStemmer for stemming and WordNetLemmatizer for lemmatization.

Model	Top 10 Aligned Tokens									
GPT-Neo	_and	,	Ċ	_in	_(	.	_the	_as	_on	_for
SGPT_{nli}	_2003	2003	_03	_3	_March	_game	_released	_three	_games	03
$\text{SGPT}_{\text{msmarco}}$	_Advance	_Game	_Released	_Releases	_ADV	Game	_GAME	_release	_released	_releases
OPT	Ċ	_The	_It	_A	_In	_This	</s>	_An	_As	_Its
OPT_{EOL}	released	Re	Released	reve	Game	re	November	It	in	In
$\text{OPT}_{\text{EOL+CSE}}$	_Game	_March	_games	_Nintendo	_game	_Microsoft	_PlayStation	_Games	Game	_2003
LLaMA	<0x0A>	_The	_It	_A	_In	_This	_Play	_An	_As	</s>
$\text{LLaMA}_{\text{EOL}}$	Re	it	re	It	_Re	_it	_It	in	The	In
$\text{LLaMA}_{\text{EOL+CSE}}$	_game	_games	_Game	game	Game	_Games	_March	_release	_released	_November
Mistral	,	_and	2	_	1	_in	_(	_as	-	_the
GritLM	_Game	_Xbox	_Pok	_game	_cross	_revealed	_Windows	,	_	_reveal
LLM2Vec	_release	_releases	_released	_Release	_revealed	_releasing	release	_Xbox	_game	_reveal

Table 2: The top 10 aligned tokens for eight $\hat{f}$ for text embedding and their corresponding f for text generation when the input text is "*Revealed in March 2003, it was released across Game Boy Advance, PlayStation 2, GameCube, Xbox and Microsoft Windows in November 2003*".

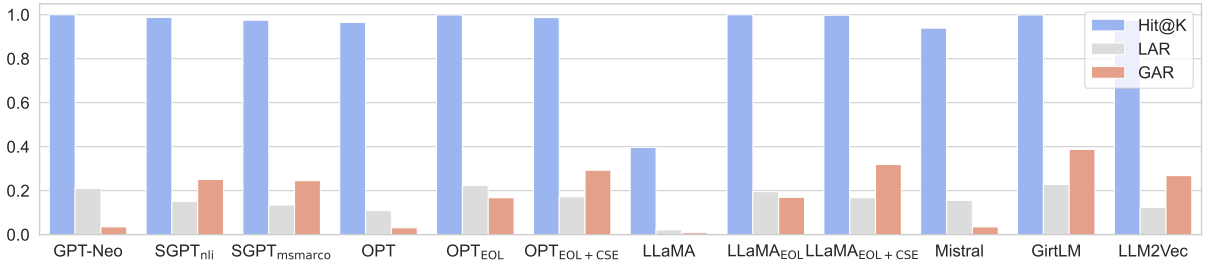


Figure 2: The comparison of evaluation metric when embedding with eight embedders $\hat{f}$ and their corresponding f.

kens, such as "and" and "the" etc; (2) compared to those aligned from f, the text embeddings from $\hat{f}$ also align with the tokens related to T_{s_i} but more meaningful, such as "game" and "November"; (3) even though some tokens are marked red, this only means that they are literally unrelated to T_{s_i}, but there may be a deeper connection. For example, "Nintendo" is the development company of "Game Boy Advance" in the input text.

Quantitative Study To quantitatively reflect the connection between $\hat{T}^K_{s_i}$ and T_{s_i}, we propose three evaluation metrics:

Hit@K To measure whether the top K tokens of $\hat{T}_{s_i}$ contains any token in T_{s_i}, we propose the metric of Hit@K as follows:

$$\text{Hit@K} = \mathop{\mathbb{E}}_{s_i \in D} \left[\mathbb{I} \left(\left| \hat{T}^K_{s_i} \cap T_{s_i} \right| > 0 \right) \right] \quad (6)$$

where $\mathbb{I}(.)$ is the indicator function, $|\cdot|$ represents the element number of the set.

Local Alignment Rate To measure the overlap degree between the tokens in T_{s_i} and the top $|T_{s_i}|$ tokens in $\hat{T}_{s_i}$, we propose the metric of Local Alignment Rate (LAR) as follows:

$$\text{LAR} = \mathop{\mathbb{E}}_{s_i \in D} \left[\left| \hat{T}^{K_i}_{s_i} \cap T_{s_i} \right| / K_i \right] \quad (7)$$

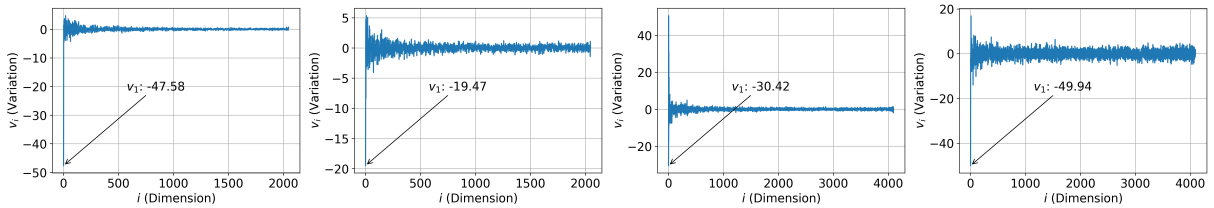
where K_i is denoted as $|T_{s_i}|$ for simplicity.

Global Alignment Rate LAR can not reflect the global alignment situation. For example, elements in $\hat{T}^{K_i}_{s_i} \cap T_{s_i}$ and $\hat{T}^{K_j}_{s_j} \cap T_{s_j}$ can be either the completely same or completely different, but cannot be reflected in LAR. To measure the overlap degree in the dataset D globally, we propose the metric of Global Alignment Rate (GAR) as follows:

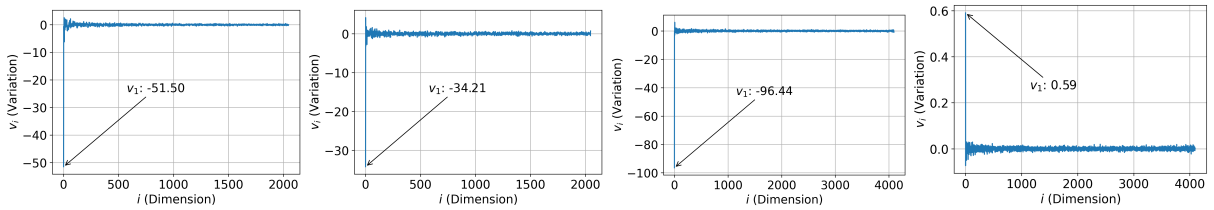
$$\text{GAR} = \left| \cup_{i=1}^{|D|} \left(\hat{T}^{K_i}_{s_i} \cap T_{s_i} \right) \right| / \left| \cup_{i=1}^{|D|} T_{s_i} \right| \quad (8)$$

where $|D|$ represents the text number of D.

We report the Hit@10, LAR, and GAR for all embedders $\hat{f}$ and their corresponding f used for



(a) GPT-Neo → SGPT_{nli} (b) OPT → OPT_{EOL} (c) LLaMA → $\text{LLaMA}_{\text{EOL}}$ (d) Mistral → LLM2Vec



(e) GPT-Neo → $\text{SGPT}_{\text{msmarco}}$ (f) OPT → $\text{OPT}_{\text{EOL+CSE}}$ (g) LLaMA → $\text{LLaMA}_{\text{EOL+CSE}}$ (h) Mistral → GritLM

Figure 3: The variation in each principal component of the embedding space.

text embedding in Figure 2. The following findings can be easily concluded: (1) all f and $\hat{f}$ except LLaMA maintain a high Hit@10, which means at least one token in the input text is aligned; (2) all $\hat{f}$ also maintain a low LAR and but higher GAR than that of the corresponding f; (3) compared to OPT_{EOL} and $\text{LLaMA}_{\text{EOL}}$, $\text{OPT}_{\text{EOL+CSE}}$ and $\text{LLaMA}_{\text{EOL+CSE}}$ lead to a lower LAR and a higher GAR after contrastive learning.

Combined with the qualitative analysis, we conclude that text embeddings from f and $\hat{f}$consistently align certain tokens in the text and that $\hat{f}$-aligned tokens tend to be more diverse and more meaningful to the input text.

4 Spectral Analysis of Embedding Space

For a deeper understanding of the phenomenon, we analyze the singular value spectrum of the embedding space before and after training. Specifically, we use the same text dataset D in Section 3 and some $(f, \hat{f})$ pairs, while all texts in D are converted into embeddings via f and use the SVD decomposition to obtain a set of standard orthogonal bases in d-dimensional space, which can be expressed as

$$\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_d] \in \mathbb{R}^{d \times d} \tag{9}$$

where $\mathbf{u}_j \in \mathbb{R}^{d \times 1}$ corresponds to the singular vector of j-th largest singular value.

For any text s_i from D, we denote its embedding obtained from f and $\hat{f}$ as $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$, separately. Then we metric the variation in each principal component between $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$ based on $\mathbf{U}$:

$$v_j = \mathop{\mathbb{E}}_{s_i \in D} \left[\left(\hat{\mathbf{h}}_i - \mathbf{h}_i \right)^\top \mathbf{u}_j \right] \tag{10}$$

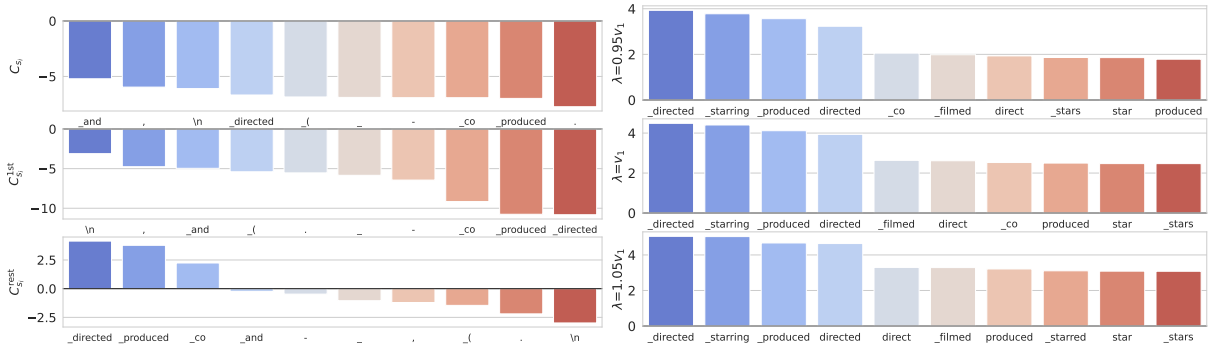
where v_j represents the variation in the j-th largest principal component. Due to space limitations, we select four $(f, \hat{f})$ pairs and plot their $\{v_j\}_{j=1}^{d}$ in Figure 3. Then we have the observation as belows:

Observation 1. *Compared to the original embedding space, the variation of the first principal component, i.e., v_1, is dominant.*

Specifically, compared with the original LLMs, the embedding spaces of the most $\hat{f}$ models decrease significantly on the first principal component. Two special cases are $\text{LLaMA}_{\text{EOL}}$ and GritLM: (1) $\text{LLaMA}_{\text{EOL}}$varies greatly in each of the first few principal components. We conjecture that the anomalies of $\text{LLaMA}_{\text{EOL}}$ indicate precisely that its embedding space is not good enough. It is corroborated by the fact that $\text{LLaMA}_{\text{EOL+CSE}}$ in Figure 3 behaves consistently with other embedders; (2) GritLM shows a small increase in the principal component. We speculate that this results from co-tuning with contrastive learning and next-token prediction. It is corroborated by the behavior of the same Mistral-based LLM2Vec, which is fine-tuned with contrastive learning only and has a decrease in the first principal component.

We further analyze the contribution of the first principal component and the other components in aligning tokens. Specifically, we divide the text embedding $\mathbf{h}_i$ into two components:

$$\mathbf{h}_i = \mathbf{h}_i^{\text{1st}} + \mathbf{h}_i^{\text{rest}} \tag{11}$$



(a) Contribution to the aligned tokens.

(b) Aligned tokens after adjusting $\mathbf{u}_1$.

Figure 4: The situation of the aligned token when f is GPT-Neo, $\hat{f}$ is SGPT_{nli}, and the input text is "*Making a Killing is a 2018 Canadian-American crime-mystery film co-written, co-produced and directed by Devin Hume.*"

where $\mathbf{h}_i^{\text{1st}} = \mathbf{u}_1^\top \mathbf{h}_i \mathbf{u}_1$ and $\mathbf{h}_i^{\text{rest}} = \sum_{j=2}^{d} \mathbf{u}_j^\top \mathbf{h}_i \mathbf{u}_j$. We then measure the contribution of $\mathbf{h}_i^{\text{1st}}$ and $\mathbf{h}_i^{\text{rest}}$ to aligning tokens. Based on the matrix decomposition, we divide the contribution into two parts:

$$\underbrace{\mathbf{E}_g \mathbf{h}_i}_{C_{s_i}} = \underbrace{\mathbf{E}_g \mathbf{h}_i^{\text{1st}}}_{C_{s_i}^{\text{1st}}} + \underbrace{\mathbf{E}_g \mathbf{h}_i^{\text{rest}}}_{C_{s_i}^{\text{rest}}}. \tag{12}$$

Specifically, we sample a text s_i from D, rank and obtain the top K tokens based on C_{s_i} and see how much $C_{s_i}^{\text{1st}}$ and $C_{s_i}^{\text{text}}$ contribute to the logits. Due In Figure 4a, we provide an example and obtain the following observation:

Observation 2. *The first principal component contributes much more to meaningless tokens than meaningful tokens.*

Combining Observation 1 and 2, we can see: (1) current text LLM-based embedders always maximize the perturbation of the first principal component, while (2) the first principal component contributes mainly to meaningless tokens. Therefore, we give the following hypothesis:

Hypothesis 1. *The text embeddings of original LLMs have been aligned with the key tokens but are not reflected due to the affection by the first principal component.*

To verify the hypothesis, we manually adjust the embeddings from f. Specifically, considering that the variation on the other principal components is small compared to the first principal component, we can simplify as follows:

$$\begin{aligned} &\mathop{\mathbb{E}}_{s_i \in D} \left[\left(\hat{\mathbf{h}}_i - \mathbf{h}_i \right)^\top \mathbf{U} \right] \approx [v_1, 0, \cdots, 0] \\ \Rightarrow &\mathop{\mathbb{E}}_{s_i \in D} \hat{\mathbf{h}}_i \approx \mathop{\mathbb{E}}_{s_i \in D} \mathbf{h}_i + v_1 \mathbf{u}_1 \end{aligned} \tag{13}$$

Therefore, for each text embedding $\mathbf{h}_i$, we subtracted a certain amount of the first principal component and obtained the adjusted embedding $\mathbf{h}_i^{\text{adj}}$:

$$\mathbf{h}_i^{\text{adj}} = \mathbf{h}_i + \lambda \mathbf{u}_1 \tag{14}$$

where $\lambda \in R$ is a hyper-parameter. In Figure 4b, we report the top 10 tokens aligned by $\mathbf{h}_i^{\text{adj}}$ and their corresponding logits when adjusting λ for $0.95v_1$, v_1 and $1.05v_1$. As shown in Figure 4b, the embedding from f can align with more meaningful tokens of the input text by adjusting only the first principal component, verifying our hypothesis. The similar conclusions are shown on f of other studies.

5 Potential Application

5.1 Training-Free Embedding Sparsification

The LLM-based embedders show superior Information Retrieval (IR) performance over the embedding models based on Transformer encoder-only PLMs (e.g., BERT (Kenton and Toutanova, 2019) and RoBERTa (Liu et al., 2019)). However, the dimensionality of these LLMs' output embeddings (2048∼4096) far exceeds the dimensionality of BERT and RoBERTa (768∼1024), which will incur exponential computation and storage overhead in practice. To overcome this problem, we propose a new sparse retrieval method to generate high-quality query extensions for queries and sparse representations for documents.

For each document d_i, we obtain its embedding $\hat{\mathbf{h}}_{d_i}$ and aligned token set $\hat{T}_{d_i}$ using the embedding LLM. Then we can maintain a vocabulary-length sparse vector $\tilde{h}_{d_i} = [w_{t_1}, \cdots, w_{t_L}]$, where only those dimensions corresponding to the top K

aligned tokens are not zero:

$$w_{t_i} = \begin{cases} \mathbf{e}_{t_i}^{\top} \hat{\mathbf{h}}_{d_i} & \text{if } t_i \in \hat{T}_{d_i}^{K} \\ 0 & \text{otherwise} \end{cases} \quad (15)$$

For each query q_i, we get its literal token set T_{q_i} using the tokenizer and its aligned token set $\hat{T}_{q_i}$. It is easy to see that we can extend T_{q_i} using the first M elements in $\hat{T}_{q_i}$, obtaining the expanded token set $\tilde{T}_{q_i} = T_{q_i} \cup \hat{T}_{q_i}^{M}$.

In ad-hoc retrieval scenarios, all document sparse representations can be computed and cached in advance while the query is computed and extended on the fly. Therefore, we can calculate the similarity of q_i and d_j as follows:

$$\text{Similarity}(q_i, d_j) = \sum\nolimits_{t_k \in (\tilde{T}_{q_i} \cap \hat{T}_{d_i}^{K})} w_{t_i} \quad (16)$$

We select LLM2Vec and GritLM due to their SOTA performance but up to 4096 embedding dimensions. For evaluation, we select four information retrieval datasets: FiQA (Maia et al., 2018), NFCorpus (Boteva et al., 2016), SciFact (Wadden et al., 2020) and ArguAna (Wachsmuth et al., 2018) and report the nDCG@10. For hyperparameter, we experiment under the settings $K \in \{1000, 2000, 3000\}$ and $M \in \{25, 50, 75, 100\}$ and report the best results in Table 3. In the most datasets, the performance is insensitive to K, while increasing with the increase of M.

Model	FiQA	NFCorpus	SciFact	ArguAna
BM25	0.236	0.325	0.665	0.315
SPLADEv2	0.336	0.334	0.693	0.479
LLM2Vec	0.531	0.393	0.789	0.575
to Spar.	0.404	0.326	0.669	0.481
GirtLM	0.600	0.409	0.792	0.632
to Spar.	0.457	0.336	0.703	0.526

Table 3: The performance on four IR datasets. "to Spar." expresses our sparse retrieval method.

Our sparse retrieval approach preserves 80% of the text embeddings' performance, outperforming the strong baselines: BM25 and SPLADEv2. Since the length of sparse representation is fixed, our sparse retrieval method can achieve a retrieval efficiency similar to that of BM25 when ignoring the consumption of the query encoding process. Compared to the original dense retrieval method, our method only needs ∼13% FLOPs in the inference stage, with plenty of room for further improvement.

Setting	Top 5 aligned token of S_A
-wo I	_Movie _movie _cinema _movies _watched
-w I	_Joy _joy _happiness joy _Love
	Top 5 aligned token of S_B
-wo I	_movie _Movie _movies _cinema _Mov
-w I	_sad _Sad _disappointment _disappointed _anger
	Top 5 aligned token of S_C
-wo I	_afternoon _cinema _movie _Movie _movies
-w I	_joy _Joy joy _happiness _delight

Table 4: Comparison of the aligned tokens with / without the instructions prefix.

5.2 Explain Instruction-Following Capability

Recent works such as Instructor (Su et al., 2023) and InBedder (Peng et al., 2024) use different instruction prefixes to distinguish different embedding tasks. To explain how the instruction-following embedder works, we show that the same text will align to different key tokens when prompted by the task-specific instruction. Considering a toy example of three sentences: (S_A, S_B, S_C) and one instruction I:

S_A: I really enjoyed the movie last night.
S_B: I didn't enjoy the movie last night at all.
S_C: I had a great time watching the film this afternoon.
I: Classify the emotion expressed in the given Twitter message into one of the six emotions: anger, fear, joy, love, sadness, and surprise.

where I is introduced by Wang et al. (2023) and used for EmotionClassification (Saravia et al., 2018). We use LLM2Vec as the embedder and observe if aligned tokens from the same text differ with the instruction and without the instruction.

As shown in Table 4, the tokens aligned by all sentences are largely changed when adding I. When I is not added, all tokens are aligned to the non-sentiment tokens. Interestingly, when I is added, S_A and S_C are mainly aligned to the tokens for positive emotions, while S_B is mainly aligned to the tokens for negative emotions. We also find the similarities among these sentences will be different:

- When no instruction is added, the embedder can only "randomly" select some key tokens to align. For all sentences, the LLM happens to both choose topic-related tokens. As a result, similarity (S_A, S_B)=`0.821` is higher than similarity (S_A, S_C)=`0.718`.

- When the instruction for sentiment classification is added, the LLM "adaptively" selects the sentiment tokens to align with. As a result, similarity $(I + S_A, I + S_B)$=0.814 become lower than similarity $(I+S_A, I+S_C)$=0.829.

5.3 Explain Semantic Relatedness / Similarity

Text embedders are fine-tuned with different datasets depending on their evaluation task. For example, the NLI datasets are often used for training when evaluating the Semantic Text Similarity (STS) task on "semantic similarity". Instead, the MS MARCO dataset is often used for training when evaluating the information retrieval task on "semantic relatedness". It is difficult to distinguish these two fuzzy concepts for a long time (Abdalla et al., 2023). Benefiting from our finding, we can intuitively understand "semantic similarity" and "semantic relatedness" by mapping the text embeddings to token space. Considering a toy example of two sentences (S_A, S_B):

S_A: I like apples. S_B: I dislike apples.

We obtain the two sentence embeddings with SGPT_{nli} and $\text{SGPT}_{\text{msmarco}}$ and obtain the aligned tokens with the decoder layer of GPT-Neo. As there is no difference between these two embedders except for the fine-tuning dataset.

As shown in Table 5, most aligned tokens of S_A are related to "apple", while there is some difference in the tokens aligned by S_B. Specifically, when SGPT_{nli} is used, tokens related to "dislike" are in the majority, whereas when $\text{SGPT}_{\text{msmarco}}$ is used, the ratio of tokens related to "dislike" and "apple" is balanced. This difference can help intuitively understand the difference between "semantic similarity" and "semantic relatedness":

- S_A and S_B are not considered to have a high degree of similarity because S_A is an affirmative while S_B is a negative sentence. SGPT_{nli} aligns the embedding of S_B to "dislike" to ensure that the embedding of the two sentences is far enough apart. Therefore, the cosine similarity given by SGPT_{nli} is only 0.419;

- S_A and S_B can be considered highly relevant because they both describe whether "I" like "apples" or not. $\text{SGPT}_{\text{msmarco}}$ aligns the embedding of S_B to both "dislike" and "apple" to ensure that the final similarity reflects their relevance. Therefore, the cosine similarity given by $\text{SGPT}_{\text{msmarco}}$ is 0.816;

Model	Top 5 aligned token of S_A
SGPT_{nli}	_apple _apples _Apple apple Apple
$\text{SGPT}_{\text{msmarco}}$	_apple _Apple Apple apple _liking
	Top 5 aligned token of S_B
SGPT_{nli}	_dislike _disliked hate _hates _apple
$\text{SGPT}_{\text{msmarco}}$	_dislike _Apple _disliked _apple Apple

Table 5: Comparison of the aligned tokens when using different fine-tuning data.

6 Related Works

Reconstructing the information of the original text from its embedding (Pan et al., 2020) has been explored primarily as a topic in privacy and security. Recently, some works have tried reconstructing the original text from text embeddings by training additional decoders. Li et al. (2023) is the first to try a single-round reduction method, while Morris et al. (2023) and Chen et al. (2024) use an iterative multi-round method, Vec2Text, to achieve better text reconstruction performance. Unlike these methods, this work does not involve any training process but only draw on the decoding layers in the LLMs.

The most related work is Ram et al. (2023), who find that embeddings from several BERT-based models align with key tokens after passing through the MLM head from the original BERT. Our work differs in three aspects: (1) Ram et al. (2023) observe this in three models, while we find that many <1B models (e.g., SimCSE (Gao et al., 2021), Contriever (Izacard et al., 2022) and E5 (Wang et al., 2022)) do not exhibit this, motivating our focus on LLMs where the phenomenon consistently holds; (2) they describe the effect, whereas we further explain its cause via spectral analysis; (3) they focus on dense retrieval, while we extend to sparse retrieval and interpretability applications.

7 Conclusion

In this work, we show the alignment of text embeddings obtained from LLMs for embedding with key tokens in the input text. We first perform qualitative and quantitative analyses on eight LLMs to demonstrate the generalizability of our conclusions. Then, we use spectral analysis to understand the phenomenon better and show that text embeddings can be aligned to key tokens by adjusting the first principal component. For application, three examples given on information retrieval and interpretability demonstrate our findings' broad application promise and continued research value.

Limitation

We summarize the limitations as follows:

- For universality, we cannot observe a similar phenomenon in the encoder-only PLM-based embedders except for several special cases. We conjecture that the reason comes from two sources: (1) encoder-only PLMs have a larger variation in the embedding space than LLMs due to too few parameters; (2) encoder-only PLMs use a complex MLM head for training, and the text embedding is obtained too far away from the final decoded token embedding matrix, resulting in no dependencies between them.

- For the LLM-based embedders, we only conducted the empirical study for the LLMs for English embedding. We have not extended the study to a multi-lingual setting due to insufficient LLMs for multi-lingual embedding.

- In Section 4, we have only shown that adjusting the first principal component can achieve alignment with key tokens, but we are unable to explain why the LLMs' pre-training phase leads to such an embedding space, nor can we achieve the same performance as the existing methods by tuning only the first principal component. At the same time, it is conceivable that we cannot achieve a similar embedding quality to contrastive learning by adjusting only the first principal component.

Acknowledgements

This work was supported by the National Science and Technology Major Project under Grant 2022ZD0120202, in part by the National Natural Science Foundation of China (No. U23B2056), in part by the Fundamental Research Funds for the Central Universities, and in part by the State Key Laboratory of Complex & Critical Software Environment.

References

Mohamed Abdalla, Krishnapriya Vishnubhotla, and Saif Mohammad. 2023. What makes sentences semantically related? a textual relatedness dataset and empirical study. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 782–796.

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

Vera Boteva, Demian Gholipour, Artem Sokolov, and Stefan Riezler. 2016. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*, pages 716–722. Springer.

Samuel Bowman, Gabor Angeli, Christopher Potts, and Christopher D Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Yiyi Chen, Heather Lent, and Johannes Bjerva. 2024. Text embedding inversion security for multilingual language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7808–7827.

Guy Dar, Mor Geva, Ankit Gupta, and Jonathan Berant. 2022. Analyzing transformers in embedding space. *arXiv preprint arXiv:2209.02535*.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, et al. 2021. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 1:1.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. *arXiv preprint arXiv:2104.08821*.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *arXiv preprint arXiv:2203.14680*.

Edward J Hu, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2021. Lora: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

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

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023. Scaling sentence embeddings with large language models. *arXiv preprint arXiv:2307.16645*.

Jacob Devlin Ming-Wei Chang Kenton and Lee Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of naacL-HLT*, volume 1, page 2.

Yibin Lei, Di Wu, Tianyi Zhou, Tao Shen, Yu Cao, Chongyang Tao, and Andrew Yates. 2024. Meta-task prompting elicits embedding from large language models. *arXiv preprint arXiv:2402.18458*.

Haoran Li, Mingshi Xu, and Yangqiu Song. 2023. Sentence embedding leaks more information than you expect: Generative embedding inversion attack to recover the whole sentence. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 14022–14040.

Xianming Li and Jing Li. 2024. Bellm: Backward dependency enhanced large language model for sentence embeddings. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

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

Edward Loper and Steven Bird. 2002. Nltk: The natural language toolkit. In *Proceedings of the ACL-02 Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, pages 63–70.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. 2018. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942.

John Morris, Volodymyr Kuleshov, Vitaly Shmatikov, and Alexander M Rush. 2023. Text embeddings reveal (almost) as much as text. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12448–12460.

Niklas Muennighoff. 2022. Sgpt: Gpt sentence embeddings for semantic search. *arXiv preprint arXiv:2202.08904*.

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

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.

Xudong Pan, Mi Zhang, Shouling Ji, and Min Yang. 2020. Privacy risks of general-purpose language models. In *2020 IEEE Symposium on Security and Privacy (SP)*, pages 1314–1331. IEEE.

Letian Peng, Yuwei Zhang, Zilong Wang, Jayanth Srinivasa, Gaowen Liu, Zihan Wang, and Jingbo Shang. 2024. Answer is all you need: Instruction-following text embedding via answering the question. *arXiv preprint arXiv:2402.09642*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Ori Ram, Liat Bezalel, Adi Zicher, Yonatan Belinkov, Jonathan Berant, and Amir Globerson. 2023. What are you token about? dense retrieval as distributions over the vocabulary. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2481–2498.

Elvis Saravia, Hsien-Chi Toby Liu, Yen-Hao Huang, Junlin Wu, and Yi-Shin Chen. 2018. Carer: Contextualized affect representations for emotion recognition. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pages 3687–3697.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition improves language model embeddings. *arXiv preprint arXiv: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*, pages 1102–1121.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. 2018. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. 2020. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550.

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. *arXiv preprint arXiv:2212.03533*.

Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2023. Improving text embeddings with large language models. *arXiv preprint arXiv:2401.00368*.

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. *arXiv preprint arXiv:2404.03921*.