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

Anonymous EMNLP submission

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 embedding LLMs, the obtained text embedding will be able to be aligned with the key tokens in the input text. We first fully analyze this phenomenon on eight embedding LLMs and show that this phenomenon is universal and is not affected by model architecture, training strategy, and embedding method. With a deeper analysis, we then find that the main change in embedding space between the embedding LLMs and their original generative LLMs is in the first principal component. By adjusting the first principal component, we can align text embedding with the key tokens. Finally, we give several examples to demonstrate the vast 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 fresh perspective to help understand fuzzy concepts (e.g., semantic relatedness vs. semantic similarity) and emerging technologies (e.g., instruction-following embedding) in this field.

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., 2023b; Springer et al., 2024; BehnamGhader et al., 2024).

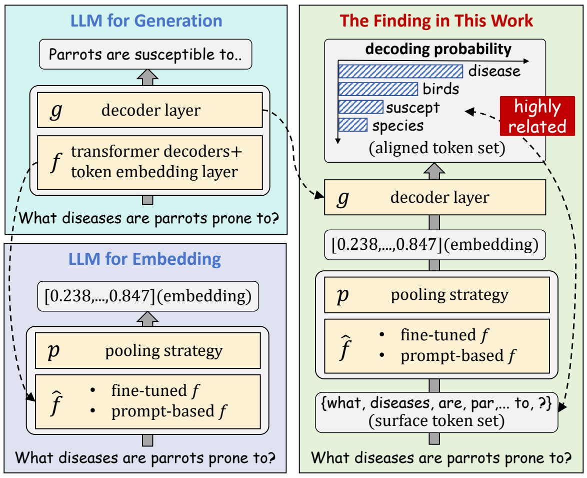


Figure 1: Existing paradigms on LLMs for text generation and embedding (left) and the novel findings of this work (right).

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., 2023b; Springer et al., 2024) or contrastive learning (Muennighoff, 2022; BehnamGhader et al., 2024); and (3) a pooling strategy p is used to weighted 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 surrounds a very interesting finding: when the text embedding obtained by $\hat{f}$ passes through the decoder 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.

Considering the unusual nature of this phenomenon, we first introduce eight LLMs for text embedding and prove that the above phenomenon is universal and independent of the LLMs' architecture, the training strategy, and the embedding method. Subsequently, we performed qualitative and quantitative analyses based on these LLMs to understand this finding more intuitively and precisely. (**Section 3**). To better explain this phenomenon, we compare the embedding spaces of f and $\hat{f}$ using spectral analysis (**Section 4**). We find that the dominant change in $\hat{f}$ is mainly 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 (**Section 5**). For example, we find that the criticism of LLM-generated embedding mainly stems from its high dimensionality (1024-4096), 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's dense retrieval and outperforms strong baselines like BM25 (Robertson et al., 2009) and SPLADE v2 (Formal et al., 2021). At the same time, we show that our work helps to intuitively understand (1) the training-data influence to semantic relevance and semantic similarity tasks and (2) the working mechanism of the instruction-following embedding (Su et al., 2023) in the Appendix.

The contributions of this paper are summarized as follows:

- We find an interesting and unusual phenomenon: the text embeddings obtained in the embedding LLM align with the key tokens;
- 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 example applications, including improvements to the method and interpretability of the model, demonstrating that our findings have large application value.

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 an encoder. 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 column vector of $\mathbf{H}$. 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 α_j is the weight of the hidden states and $\sum_{j=1}^{L} \alpha_j = 1$. Specifically, there are three popular pooling strategies in practice: for last pooling, $\alpha_j = 1$ when $j = L$ else is 0; for mean pooling, $\alpha_j = 1/L$ for each i; 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, i.e., the next token prediction, is not designed for embedding, and the unidirectional attention detracts from the expressive power of the hidden states (Springer et al., 2024). In the subsequent subsections, we describe how the existing methods improve the embedding's quality based on the top of f. For simplicity, we indiscriminately refer to the models proposed by the existing methods as $\hat{f}$.

2.2 Embedding via Prompt Engineering

The model $\hat{f}$ 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., 2023b) 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 the better prompt template based on task-oriented (Lei et al., 2024) or chain-of-thought (Zhang et al., 2024) can lead to better performance. Springer et al. (2024) proposes a prompt template: `Rewrite the sentence: [text], rewritten sentence: [text]`, where both `[text]` are the placeholder. In practice, both placeholders are filled with the same text, and the text embedding is obtained by the mean pooling strategy, but it is pooled only within the range of the second occurrence of the text.

The methods based on prompt engineering are simple and training-free, so they do not potentially compromise the generative capabilities of the LLMs. However, they provide limited performance improvement for text embedding tasks.

2.3 Embedding via Contrastive Learning

The methods based on contrastive learning inherited the good 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 sampled from 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 labelled text pair, which can be query-document, question-answer or hypothesis-entailment (Li et al., 2023), etc., while potential hard negative pairs may be introduced, such as hypothesis-contradiction. 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 $d(.,.)$ 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) finds that multiplying the feed-forward value vector with the token embedding matrix can obtain a distribution over the vocabulary, and the tokens with high probability can explain what the FFN updates to hidden layer representations. Inspired by these works, we try to interpret the text embeddings obtained from LLMs by mapping them into the token space.

3.2 Method

To implement the above idea, we need a text dataset D, and some $(\hat{f}, T, \mathbf{E}_g)$ triplets. $\hat{f}$ is the LLM output d-dimensional text embeddings, $T = \{t_1, \cdots, t_L\}$ is the L-sized vocabulary and $\mathbf{E}_g = [\mathbf{e}_{t_1}, \cdots, \mathbf{e}_{t_L}]^\top \in \mathbb{R}^{L \times d}$ 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. $\mathbf{E}_g$ is also the only parameter in g[1],

[1]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	Task	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$\rightarrow$SCL	E5	weighted mean	cosine

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

therefore, there is no difference between $\mathbf{E}_g\mathbf{h}_i$ and $g(\mathbf{h}_i)$ for any text embedding $\mathbf{h}_i \in \mathbb{R}^{d\times 1}$.

Process 1 Embedding-Token Alignment Analysis

Input: A text dataset D and the $(\hat{f}, T, \mathbf{E}_g)$ triplet.
1: Initialization: $i \leftarrow 0, j \leftarrow 0$
2: **while** $i \leq |D|$ **do**
3: Get the i-th text s_i in D
4: Deduplicate tokenizer(s_i) to obtain T_{s_i}
5: Calculate $\mathbf{h}_i \leftarrow$ pooling$(\hat{f}(s_i))$
6: **while** $j \leq |T|$ **do**
7: Calculate $p(t_j|s_i) \leftarrow \mathbf{e}_{t_j}^\top \mathbf{h}_i$
8: Update $j \leftarrow j + 1$
9: **end while**
10: Sort T in descending order by $p(t_i|s_i)$ to get $\hat{T}_{s_i}$
11: Update $i \leftarrow i + 1$
12: **end while**
Output: T_{s_i} and $\hat{T}_{s_i}$

Given a text s_i sampled from D, we need to obtain its literal token set T_{s_i} and aligned token set $\hat{T}_{s_i}$ and capture the potential relation between these two sets. We use Process 1 to analyze the alignment of text embedding with the tokens. 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 $\hat{T}_{s_i}$, we (1) follow the pooling strategy of $\hat{f}$ to obtain the text embedding, (2) multiply the text embedding with the token embedding matrix and get the decoding score $p(t_j|s_i)$ for each token t_j, and (3) obtain the ordered token set $\hat{T}_{s_i}$ by sorting in descending order according to the score.

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. We observe that experiments on other datasets, such as SNLI (Bowman et al., 2015) and MSMARCO (Nguyen et al., 2016), lead to similar conclusions; please refer to Appendix E for details.

Triplet $(\hat{f}, T, \mathbf{E}_g)$ We selected eight embedding model based on LLMs for analysis, which includes 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., 2023b); 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 embedding LLMs as $\hat{f}$ and obtain T and $\mathbf{E}_g$ from their backbone models.

Note that none of the improvement ideas for these models go beyond what we describe in Section 2, and please refer to Appendix A for detailed information on each model. Additionally, to ensure the generalizability of subsequent conclusions, the LLMs selected have the following attributes:

- **Different Architecture:** The backbones of these LLMs include: GPT-Neo (Gao et al., 2020), OPT (Zhang et al., 2022), LLaMA (Touvron et al., 2023) and Mistral (Jiang et al., 2023a). GritLM and LLM2Vec enable bi-directional attention, while the other LLMs keep the casual attention;
- **Different Fine-Tuning Methods:** These LLMs rely on different methods to improve the embedding capability, such as prompt engineering, contrastive learning, or multi-task learning, while the different corpus was used for fine-tuning.
- **Different Embedding Methods:** These LLMs use different pooling strategies to obtain embeddings and calculate the similarity using cosine similarity or dot product [2].

[2] Regardless of what the similarity metric is recommended,

Model	Top 10 Aligned Tokens									
GPT-Neo	_and	,	Ċ	_in	_(	.	_the	_as	_on	_for
SGPT$_{\text{nli}}$	_2003	2003	_03	_3	_March	_game	_released	_three	_games	03
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$_{\text{EOL}}$	released	Re	Released	reve	Game	re	November	It	in	In
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>
LLaMA$_{\text{EOL}}$	Re	it	re	It	_Re	_it	_It	in	The	In
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*".

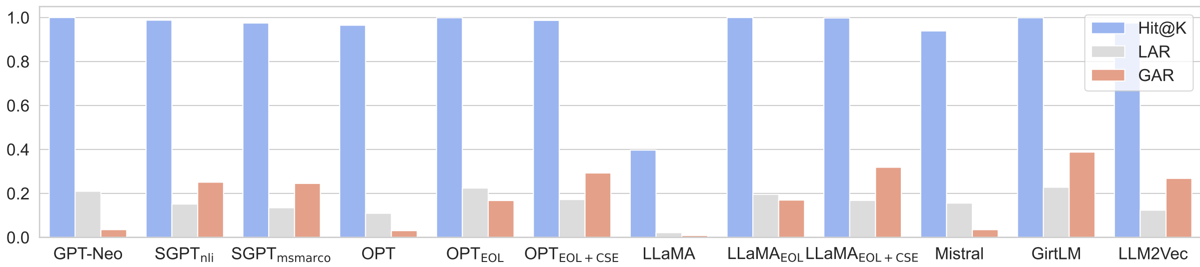


Figure 2: The evaluation metric comparison of four LLMs and their eight variant for text embeddings.

3.4 Qualitative Analysis

Since the ordered set $\hat{T}_{s_i}$ is as large as T, we analyze only the top K tokens in $\hat{T}_{s_i}$. We introduce $\hat{T}^K_{s_i}$ to denote the first K elements in $\hat{T}_{s_i}$. 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.

We use different colours to indicate the relationship between each token and the surface token set T_{s_i}: Green represents the token is in T_{s_i}; Yellow represents the token and a token in T_{s_i} are same after stemming or lemmatization[3]; Red represents the token and any 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 tokens, 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. Note that the input text is not specially selected, and we provide more cases in Appendix E.

3.5 Quantitative Analysis

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:

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

we use a simple matrix multiplication between $\mathbf{E}_g$ and $\mathbf{h}_i$, to ensure consistency with the original decoding process.

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

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:

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

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 $\hat{T}_{s_i}^{K_i} \cap T_{s_i}$ and $\hat{T}_{s_j}^{K_j} \cap T_{s_j}$ 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:

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

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

We report the Hit@10, LAR, and GAR for the original LLM and their variants used for 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 aligns certain tokens in the text, and that $\hat{f}$-aligned tokens tend to be more diversed and more key to the input text.

3.6 Discussion

How to understand? The text embedding aligns well with some key tokens in the input text after passing through the decoder layer, which means the text embedding is closer to these tokens than other tokens in high-dimensional space. Note that **the absolute position of the text embedding in the whole space** is described here, rather than in a subspace, since, as far as we can observe, all LLMs' decoder layer weights, i.e., the token embedding matrixes, are full rank.

How to explain? The explanation for this phenomenon is not straightforward because (1) the decoding layer, whose weights are never seen during the process from f to $\hat{f}$, can precisely decode some tokens related to the input text from the embedding; (2) the optimization objective of contrastive learning by itself does not guarantee that this will happen. Therefore, we analyze the singular value spectrum of the embedding space before and after training in Section 4.

How to use? This interesting finding brings extreme interpretability to text embedding. In Section 5, we show the aligning tokens of the text embedding change with different training data and different instructions. Meanwhile, we propose a sparse retrieval method for solving the computational and storage overhead caused by the ultra-high dimensionality of LLM representations.

4 Spectral Analysis of Embedding Space

For a deeper understanding of the phenomenon, we use the same text dataset D in Section 3 and some $(f, \hat{f})$ pairs. We convert all texts in D 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.

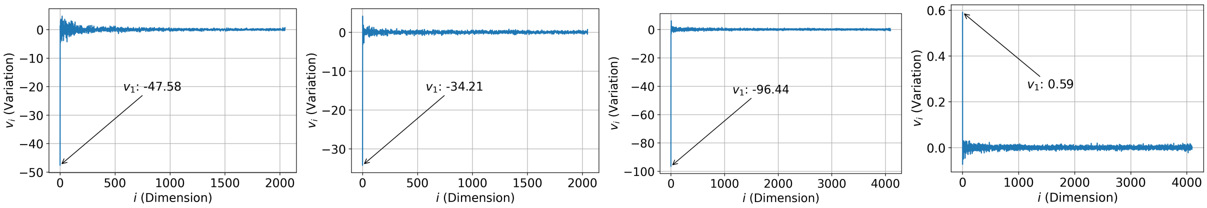
For any text s_i from D, we denote its embedding obtained from f and $\hat{f}$ as $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$, separately. Then we metric the variation in each principal component between $\mathbf{h}_i$ and $\hat{\mathbf{h}}_i$ based on $\mathbf{U}$:

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

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 and show the variation of the other four embedding LLMs in Appendix D.

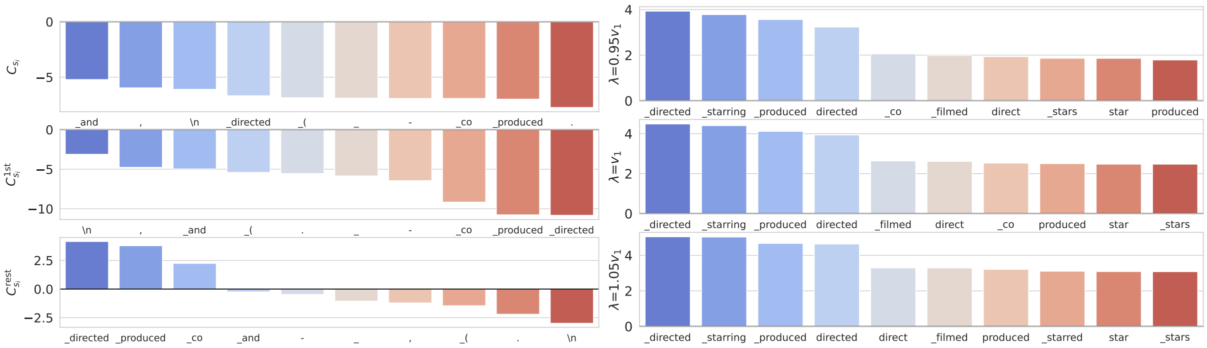
Observation 1. *Compared to the original embedding space, the variation of the largest principal component, i.e., v_1, is dominant.*

Compared with the original LLMs, the embedding space corresponding to SGPT_{nli}, $\text{OPT}_{\text{EOL+CSE}}$, and $\text{LLaMA}_{\text{EOL+CSE}}$ significantly decreases in the first principal component, while only the pair corresponding to GritLM shows a small increase. As



(a) GPT-Neo and SGPT$_{\text{nli}}$ (b) OPT and OPT$_{\text{EOL+CSE}}$ (c) LLaMA and LLaMA$_{\text{EOL+CSE}}$ (d) Mistral and GritLM

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



(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$_{\text{nli}}$ 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.*"

with the qualitative analysis, we speculate that this results from co-tuning with contrastive learning and next-token prediction. We further find that the embedding space corresponding to LLM2Vec has a significant decrease in the first principal component, too; please refer to Appendix D for details.

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:

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

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 embedding LLMs 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:

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

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 $0.95 v_1$, v_1 and $1.05 v_1$. 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. We show that similar conclusions exist on f of other studies in Appendix D.

5 Potential Application

Sparse Retrieval The LLMs for embedding show superior Information Retrieval (IR) performance over the embedding models based on traditional PLMs (e.g., BERT (Kenton and Toutanova, 2019) and RoBERTa (Liu et al., 2019)). However, the dimensionality of these LLMs' output embeddings (1024∼4096) far exceeds the 768 dimensions of traditional PLMs, 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:

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

For each query q_i, we get its surface 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:

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

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
+Spar.	0.404	0.326	0.669	0.481
GirtLM	0.600	0.409	0.792	0.632
+Spar.	0.457	0.336	0.703	0.526

Table 3: The performance comparison on the four IR datasets. "+ Spar." is our sparse retrieval method.

We select LLM2Vec and GritLM due to their SOTA performance and 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 hyper-parameter, 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. We report the detailed results in Appendix C and find that performance is insensitive to K, while increasing with the increase of M in most cases.

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.

More Insights Due to space constraints, we provide more sights in Appendix B, mainly explaining that different training data and different instructions align embeddings of the same input text to different tokens, achieving better performance for specific downstream tasks.

6 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 interpretability or information retrieval demonstrate our findings' broad application promise and continued research value.

Limitation

We summarize several limitations of this work as follows:

- For the universality of our findings, we cannot observe a similar phenomenon in the embedding models based on traditional PLMs (such as SBERT (Reimers and Gurevych, 2019) or SimCSE (Gao et al., 2021)). We conjecture that the reason comes from two sources: (1) traditional PLMs have a higher degree of embedding space variation than LLMs due to too few parameters; (2) traditional 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 study targets, 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 have not yet been able to explain why the pre-training phase of the LLMs can form 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.

References

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.

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.

Thibault Formal, Carlos Lassance, Benjamin Piwowarski, and Stéphane Clinchant. 2021. Splade v2: Sparse lexical and expansion model for information retrieval. *arXiv preprint arXiv:2109.10086*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

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*.

Minyoung Huh, Brian Cheung, Tongzhou Wang, and Phillip Isola. 2024. The platonic representation hypothesis. *arXiv preprint arXiv:2405.07987*.

Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. 2023a. Mistral 7b. *arXiv preprint arXiv:2310.06825*.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2023b. 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*.

Zehan Li, Xin Zhang, Yanzhao Zhang, Dingkun Long, Pengjun Xie, and Meishan Zhang. 2023. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*.

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.

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.

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.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

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.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

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

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of NAACL-HLT*, pages 1112–1122.

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

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

A Detailed Information of Models

This section details the text embedding LLMs used for the study in the main text. Note taht these LLMs are CC BY 4.0 compliant and open source and can be used to obtain text embedding or any of the downstream applications they support.

- **SGPT$_{nli}$** are based on GPT-Neo and fine-tuned with contrastive learning on both SNLI (Bowman et al., 2015) and MNLI (Williams et al., 2018) dataset. SGPT$_{nli}$ includes four versions of 125m, 1.3B, 2.7B and 5.7B variants, and the variant used for this work is `SGPT-1.3B-weightedmean-nli`[4].

- **SGPT$_{msmarco}$** share the same backbone and training paradigm with SGPT$_{nli}$ except the training data. The variant used for this work is `SGPT-1.3B-weightedmean-msmarco-specb-bitfit`[5].

- **OPT$_{EOL}$** are based on OPT and use prompt template `This sentence:"[text]" means in one word:"` to guide OPTs in aggregating the semantics of the whole text into a single location. Due to the training-free nature of OPT$_{EOL}$, it can be easily applied to any variant of OPT, and the variant used for this work is `OPT-1.3B`[6].

- **OPT$_{EOL+CSE}$** are parameter-efficient fine-tuned with contrastive learning on SNLI and MNLI dataset on the top of OPT$_{EOL}$. All LoRA weights of OPT$_{EOL+CSE}$ are open-sourced, and the weight corresponds to OPT-1.3B[7] are used for comparing fairly with OPT$_{EOL}$.

- **LLaMA$_{EOL}$** share the same prompt template with OPT$_{EOL}$ but are based on LLaMA. The variant used for this work is `LLaMA-7B`[8].

- **LLaMA$_{EOL+CSE}$** are parameter-efficient fine-tuned with contrastive learning on SNLI and MNLI dataset on the top of LLaMA$_{EOL}$. The weight corresponds to `LLaMA-7B`[9] are used for comparing fairly with OPT$_{EOL}$.

- **GirtLM** is fine-tuned with instruction-tuning and contrastive learning to achieve a better trade-off between the generation and embedding capabilities. `GritLM-7B`[10], whose backbone is Mistral-7B-Instruct-v0.2, is used in this work.

- **LLM2Vec** is a three-step method to adjust LLMs for text embeddings, which includes (1) changing the casual attention to bi-directional attention; (2) fine-tuning the LLM with a new task, masked next token prediction (MNTP), to adapt the LLM to use bi-directional attention; (3) fine-tuning the LLM with supervised contrastive learning to improve the embedding capability. The second-step [11] and third-step [12] LoRA weights corresponding to Mistral-7B-Instruct-v0.2 are used.

B More Application Demonstration

B.1 Semantic Relevance v.s. Similarity

Current textual embedding models are often fine-tuned with different datasets depending on their evaluation task. For example, the NLI dataset is 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 relevance". Previously, it was difficult to distinguish the embedding spaces obtained from training on different datasets, although both above settings only use contrastive learning to fine-tune. Benefiting from the "token align" phenomenon, we can now understand this phenomenon by mapping the text embeddings to token space.

We select SGPT$_{nli}$ and SGPT$_{msmarco}$ to study because there is no difference between them except for the fine-tuning dataset. Considering a toy example of two sentences (S_A, S_B):

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

We obtain the embedding of both two sentences with SGPT$_{nli}$ and SGPT$_{msmarco}$ and align the embedding to the token space with the decoder layer. As shown in Table 4, 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

[4] https://huggingface.co/Muennighoff/SGPT-1.3B-weightedmean-nli
[5] https://huggingface.co/Muennighoff/SGPT-1.3B-weightedmean-msmarco-specb-bitfit
[6] https://huggingface.co/facebook/opt-1.3b
[7] https://huggingface.co/royokong/prompteol-opt-1.3b
[8] https://llama.meta.com/llama-downloads/
[9] https://huggingface.co/royokong/prompteol-llama-7b
[10] https://huggingface.co/GritLM/GritLM-7B
[11] https://huggingface.co/McGill-NLP/LLM2Vec-Mistral-7B-Instruct-v2-mntp
[12] https://huggingface.co/McGill-NLP/LLM2Vec-Mistral-7B-Instruct-v2-mntp-supervised

the majority, whereas when SGPT$_{\text{msmarco}}$ is used, the ratio of tokens related to "dislike" and "apple" is balanced.

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

Table 4: Comparison of the aligned tokens when using different fine-tuning data.

We believe that this phenomenon can help to intuitively understand the difference between "semantic similarity" and "semantic relatedness":

- In the semantic similarity setting, S_A and S_B are not considered to have a high degree of similarity because one of them is an affirmative S_And the other is a negative sentence. SGPT$_{\text{nli}}$ aligns the embedding of S_B to "dislike" to ensure that the embedding of the two sentences is far enough apart. Therefore, the similarity of the two sentences given by SGPT$_{\text{nli}}$ is only `0.419`;
- In the semantic relevance setting, S_A and S_B can be considered highly relevant because they both describe whether "I" like "apples" or not. 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 similarity of the two sentences given by SGPT$_{\text{msmarco}}$ is `0.816`;

B.2 Instruction v.s. No-Instruction

Recent works such as Instructor (Su et al., 2023; Peng et al., 2024) use different instruction prefixes to distinguish between different embedding tasks. To explain the validity of the instruction-following embedding, we show that the same text will align to different tokens when prompted by different instructions. 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 the EmotionClassification dataset (Saravia et al., 2018). We use LLM2Vec as the studied LLM and observe whether aligned tokens from the same text differ with the instruction and without the instruction.

As shown in Table 5, the tokens aligned by all sentence largely changed when adding I. Specifically, when I is not added, all tokens are aligned to the non-sentiment tokens first. Interestingly, when I is added, S_A and S_C is mainly aligned to the tokens related to "joy", while S_B is mainly aligned to the token related to "sadness".

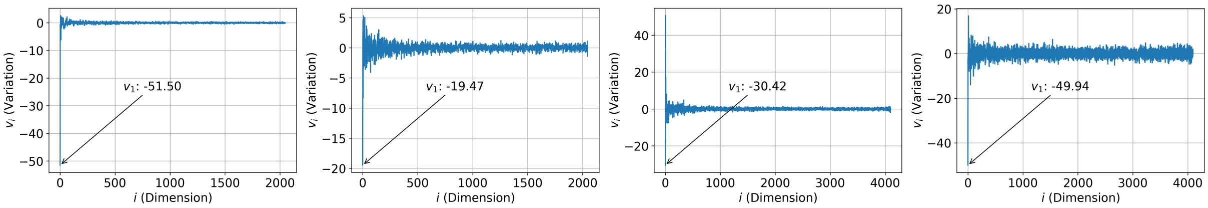
Similarly, we believe that this phenomenon can help to understand how the instruction-following embeddings work intuitively:

- When no instruction is added, the LLM can only "randomly" select some key tokens to align. For both S_A and S_B, the LLM happen 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` is lower than similarity$(I + S_A, I + S_C)$=`0.829`, .

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

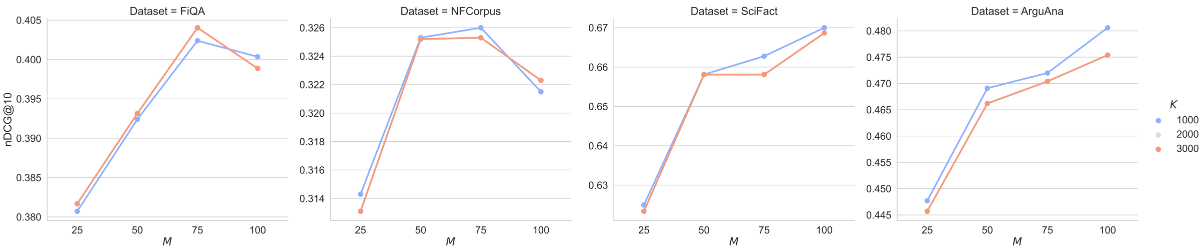
Table 5: Comparison of the aligned tokens when using the instruction or not.

Note that a similar phenomenon has also been observed by (Peng et al., 2024) under the special fine-tuning method and the last-pooling strategy. The phenomenon we observed is more general because LLM2Vec does not even seem to have any instructions when fine-tuning and is using a weighted-mean pooling strategy. We similarly emphasize that this interesting phenomenon is present in most embedding LLMs and is easy to verify.



(a) GPT-Neo and $\text{SGPT}_{\text{msmarco}}$ (b) OPT and OPT_{EOL} (c) LLaMA and $\text{LLaMA}_{\text{EOL}}$ (d) Mistral and LLM2Vec

Figure 5: The variation in each principal component of the embedding space.



(a) Performance of sparse retrieval based on LLM2Vec.

(b) Performance of sparse retrieval based on GritLM.

Figure 6: Performance Variation of sparse retrieval with hyper-parameters.

C Details of Sparse Retrieval

C.1 Evaluation Dataset and Metric

Dataset	Domain	#Query	#Corpus	Relevancy
FiQA	Finance	648	57,638	Binary
NFCorpus	Medical	323	6,633	3-level
SciFact	Science	300	5,183	Binary
ArguAna	Misc.	1,406	8,674	Binary

Table 6: Statistics of the evaluation dataset. Relevancy represents the query-document relation level.

We provide the statistics of four evaluation datasets in Table 6 and use the version provided by BEIR[13]. nDCG@10 used for evaluation is the recommended metric for the BEIR Benchmark. The calculation of nDCG@10 can be divided into two main steps: (1) calculating DCG@10:

$$DCG@10 = \sum_{i=1}^{10} \frac{2^{rel_i} - 1}{\log_2(i+1)} \quad (17)$$

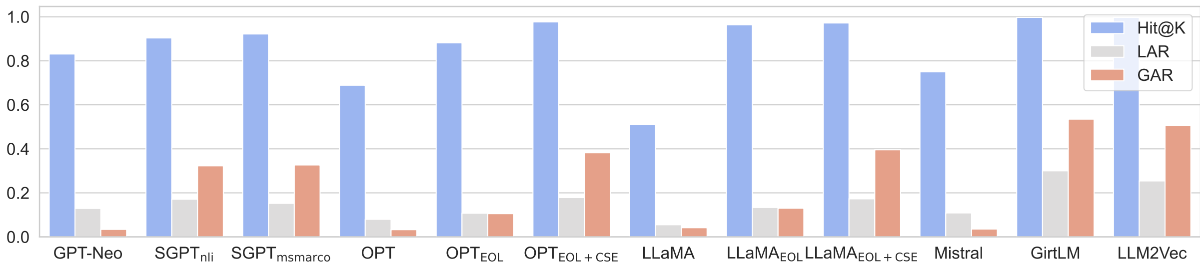
[13] https://github.com/beir-cellar/beir.

where rel_i is the relevance score of the i-th item, which is usually a nonnegative integer and $\log_2(i+1)$ is the positional discount factor, which is used to reduce the weight of lower-ranked items because users are more likely to pay attention to the top-ranked items. (2) calculating $IDCG@10$ (Ideal DCG@10), which is the DCG value when assuming that the retrieved results are ordered optimally. This means that the results are sorted from highest to lowest based on the relevance score. (3) normalizing $DCG@10$ and obtaining nDCG@10:

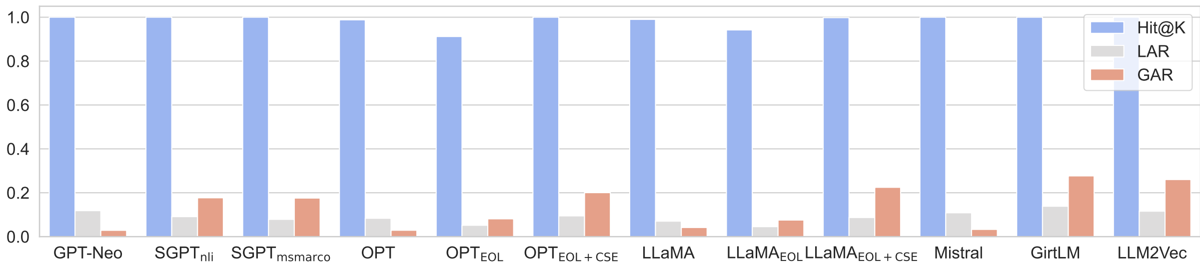
$$nDCG@10 = \frac{DCG@10}{IDCG@10} \quad (18)$$

C.2 Implementation Details

We follow the evaluation methods of LLM2Vec and GirtLM by adding different instructions in front of different datasets. The instruction is given in Table 7. We use Python 3.10 and Pytorch 2.3.0 for the implementation, while our experiments are all done on a single NVIDIA A100 40GB with CUDA 12.4.



(a) The metrics when the dataset D contains 10K documents sampled from the SNLI training set.



(b) The metrics when the dataset D contains 10K documents sampled from the MSMARCO document set.

Figure 7: The comparison of evaluation metric when embedding with eight $\hat{f}$ and their corresponding f.

C.3 Hyper-parameter Experiment

Since no validation set exists for ArguAna and Scifact, we report the performance variation on the non-zero number of sparse representation, i.e., K, and the extended token number of query, i.e., M, on the test set of all four datasets. We find that K has little effect on the results, so selecting a lower K is a good choice for low storage scenarios. The situation is more complex for M: (1) the performance on FiQA and NFCorpus peaks at M=75 while the other two datasets show a steady boost ; (2) when the embedding LLM is good enough, such as the case of GritLM, even a large M can lead to a steady boost in retrieval results.

D Additional Results on Spectral Analysis

Variation of Principal Components We show the variation principal component for the remaining 4 embedding LLMs in Figure 5. We find that, with the exception of LLaMA$_{\text{EOL}}$, the embedding spaces of the other three $\hat{f}$ decrease significantly on the first principal component. We would like to explain the anomaly of LLaMA$_{\text{EOL}}$ in terms of the recently popular "Platonic Representation Hypothesis"(Huh et al., 2024). LLaMA$_{\text{EOL}}$ is based on prompt engineering and is not considered a powerful embedding model compared to other $\hat{f}$. According to the "Platonic Representation Hypothesis", powerful embedding models always produce convergent embeddings, while weaker embedding models produce embeddings that will be more disparate from them. Thus, we conjecture that the anomalies of LLaMA$_{\text{EOL}}$ indicate precisely that the embedding space generated by it is not good enough. This is corroborated by the fact that LLaMA$_{\text{EOL+CSE}}$ in Figure 3 behaves consistently with other models.

Adjusting First Principal Components In Figure 8, we show the first principal component adjustment corresponding to the 3 additional $(f, \hat{f})$ pairs. It can be observed that although the effects vary, the overall adjusting first principal components all align the embedding to the key tokens, in line with the conclusion of Section 4.

E More Results for Analysis

E.1 Additional Qualitative Analysis

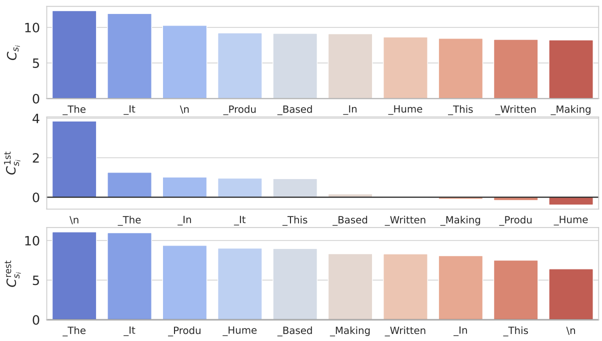
In Table 8, we provide three more examples from Wiki1M, SNLI, and MSMARCO to reflect the generalizability of our findings. We observe similar alignment phenomena as in Section 3.4, demonstrating the generalizability of our findings.

E.2 Additional Quantitative Analysis

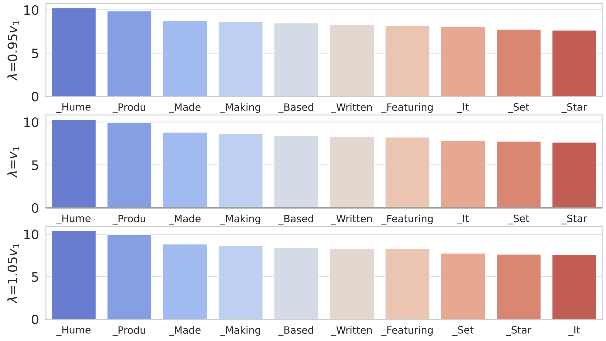
Similar to in Figure 2, we computed the same metrics on the SNLI and MAMARCO document sets and plotted the results in Figure 7. SNLI is dominated by shorter sentences, whereas MSMARCO is all about longer documents. This changes the absolute values of LAR and GAR; however, it does not affect the conclusions in Section 3.5.

Dataset	Instruction
FiQA	Given a financial question, retrieve user replies that best answer the question
NFCorpus	Given a question, retrieve relevant documents that best answer the question
SciFact	Given a scientific claim, retrieve documents that support or refute the claim
ArguAna	Given a claim, find documents that refute the claim

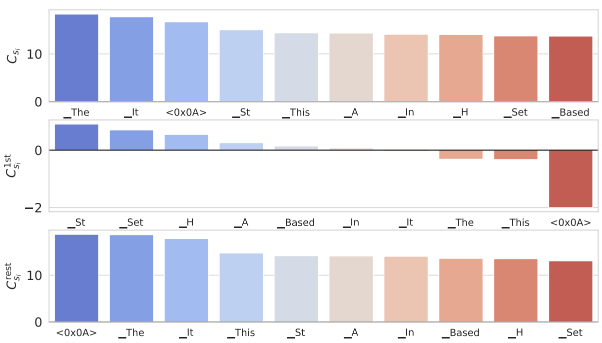
Table 7: Instruction used when obtaining embedding from LLM2Vec and GirtLM.



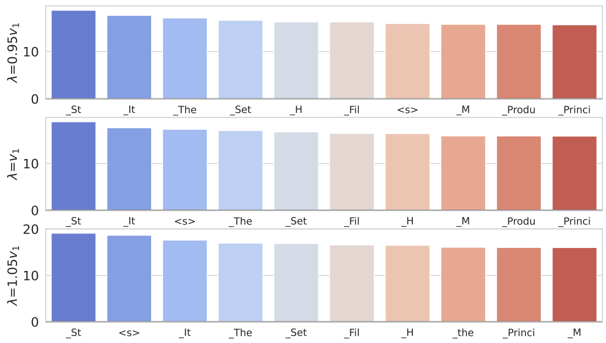
(a) Contribution to the aligned tokens (OPT).



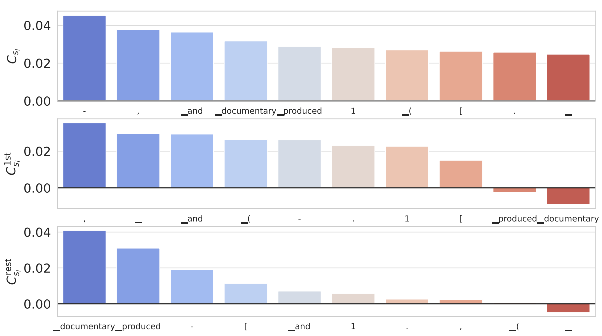
(b) Aligned tokens after adjusting $\mathbf{u}_1$ guided by $\text{OPT}_{\text{EOL+CSE}}$.



(c) Contribution to the aligned tokens (LLaMA).



(d) Aligned tokens after adjusting $\mathbf{u}_1$ guided by $\text{LLaMA}_{\text{EOL+CSE}}$.



(e) Contribution to the aligned tokens (Mistral).

(f) Aligned tokens after adjusting $\mathbf{u}_1$ guided by GirtLM.

Figure 8: Situation of the aligned token when the input text is "*Making a Killing is a 2018 Canadian-American crime-mystery film co-written, co-produced and directed by Devin Hume.*". Figure (a)-(b) show the situation when f is OPT, $\hat{f}$ is $\text{OPT}_{\text{EOL+CSE}}$; Figure (c)-(d) show the situation when f is LLaMA, $\hat{f}$ is $\text{LLaMA}_{\text{EOL+CSE}}$; Figure (e)-(f) show the situation when f is Mistral, $\hat{f}$ is GirtLM.

Dataset	Input Text
Wiki1M	*Chatwood was chosen for the role as Ubisoft wanted music that had Persian elements in it to fit the setting, while not being pure Persian music.*
GPT-Neo	_in _the , _to _" _âĢ _as _and Ċ _a
SGPT$_{nli}$	_Persian _music _MUS _Music _musical _mus _compos _Persia _Pers _Iranian
SGPT$_{msmarco}$	_Ubisoft _Music _music _MUS WOOD _soundtrack _playlist Music _XCOM _Persian
OPT	Ċ _ _The _He _Chat _It _I _This _In _They
OPT$_{EOL}$	pure not Music We _Persian Not U The music Pure
OPT$_{EOL+CSE}$	_Chat _chat _music _Music Chat _Persian _Ubisoft chat _musical _Persia
LLaMA	_The _Ch <0x0A> _He _She _" _U _It _In _This
LLaMA$_{EOL}$	Ch Pers _Pers _Ch I _we _he the The it
LLaMA$_{EOL+CSE}$	_Ch _chat _music _Pers chat _Iran Ch _musical _Music music
Mistral	, _" _in _to _for _as _and _the _a _
GritLM	_Pers _chat _Chat _wood _U _music Chat _Wood _pers Pers
LLM2Vec	_Chat _music _chat Chat _U _Pers _Music _wood _Wood chat
Dataset	**Input Text**
NLI	*In 2000, GNP was less than GDP because income receipts from the rest of the world were less than U.S. payments to the rest of the world.*
GPT-Neo	Ċ _(. , _in _G _the _GDP _and _of
SGPT$_{nli}$	_less _GN _impover _income _GDP _low _lesser _economic _little _poverty
SGPT$_{msmarco}$	_GN _GDP GN _Pik _Krugman _Gross Gs _Gn _income G
OPT	_ Ċ _In _Now _GN _The _Today _That _This _Since
OPT$_{EOL}$	GN _GN G _less less In The the _the _In
OPT$_{EOL+CSE}$	_GN _income _GDP _payments _2000 _2001 GN _Income _incomes _Global
LLaMA	\n _In _The _This _G _That _But </s> _However _Net
LLaMA$_{EOL}$	def the The _the _The In USA _U _trade G
LLaMA$_{EOL+CSE}$	_income _rece _pay _payment Rece pay _Pay _G _exports _deb
Mistral	_the , _in _(_of _for . _ _and -
GritLM	_payments _G _income _world _rece _U _rest _US _the _Pay
LLM2Vec	_G _income _payments _world _U _less _payment _rece _World _pay
Dataset	**Input Text**
MSMARCO	*Disney's Theme Parks had an operating cost of 571 million dollars divided by their 11 parks and being open 365 days a year, on average their operating cost per day is around $355,000.*
GPT-Neo	_a \n _(. _in , _for _and _the _per
SGPT$_{nli}$	Ĥ¬ _5 _five _operating _365 _$ _cost _55 _operation _operations
SGPT$_{msmarco}$	_operating _Operating _Theme _theme _operation _OPER _Operation _Parks operation _cost
GPT-Neo	_a Ċ _(. _in , _for _and _the _per
SGPT$_{nli}$	Ĥ¬ _5 _five _operating _365 _$ _cost _55 _operation _operations
SGPT$_{msmarco}$	_operating _Operating _Theme _theme _operation _OPER _Operation _Parks operation _cost
OPT	_ Ċ _So _That _This _The _They _I _If </s>
OPT$_{EOL}$	Disney _Disney $ _$ The _operating Cost average the _The
OPT$_{EOL+CSE}$	_Disney Disney _Disneyland _parks _operating _Walt _5 _costing _annual _park
LLaMA	<0x0A> 0 _This _That _The _If _Disney _With _I _In
LLaMA$_{EOL}$	Dis _Disney The the _the _ _per aver Oper oper
LLaMA$_{EOL+CSE}$	_Disney _park Theme _Park _theme park _cost _operating theme _par
Mistral	, _ 1 _in . _(_and _per _a 2
GritLM	_operating _theme _Disney _cost _Theme _day _parks _park _costs _daily
LLM2Vec	_operating _parks _park _Disney _theme _Park _Theme _Oper _daily Theme

Table 8: The top 10 aligned tokens for eight $\hat{f}$ for text embedding and their corresponding f for text generation.