CASE – Condition-Aware Sentence Embeddings for Conditional Semantic Textual Similarity Measurement

Anonymous ACL submission

Abstract

The meaning conveyed by a sentence often depends on the context in which it appears. Despite the progress of sentence embedding methods, it remains unclear as how to best modify a sentence embedding conditioned on its context. To address this problem, we propose Condition-Aware Sentence Embeddings (CASE), an efficient and accurate method to create an embedding for a sentence under a given condition. First, CASE creates an embedding for the condition using an Large Language Model (LLM) encoder, where the sentence influences the attention scores computed for the tokens in the condition during pooling. Next, a supervised method is learnt to align the LLM-based text embeddings with the Conditional Semantic Textual Similarity (C-STS) task. We find that subtracting the condition embedding will consistently improve the C-STS performance of LLM-based text embeddings and improve the isotropy of the embedding space. Moreover, our supervised projection method significantly improves the performance of LLM-based embeddings despite requiring a small number of embedding dimensions.[1]

1 Introduction

Representing the meanings of sentences using embeddings is a common NLP task (Conneau et al., 2017; Reimers and Gurevych, 2019; Gao et al., 2021; Xu et al., 2023; Chen et al., 2023) such as in Semantic Textual Similarity (STS) measurement (Agirre et al., 2012, 2015, 2016). Despite its importance, measuring STS between two sentences is a non-trivial task for humans, which is conditioned on what is being compared between the two sentences. For example, for the two sentences shown in Figure 1, human annotators would assign different similarity ratings, depending on what they are asked to compare (i.e. given conditions 1 and 2). Existing STS benchmarks do not specify the conditions under which two sentences must be compared for their semantic similarity. To address this limitation, Deshpande et al. (2023) proposed the C-STS task and a dataset, where the similarity between two sentences s_1 and s_2 is measured under two different conditions c_{low} and c_{high}, which focus on different aspects of semantics in the two sentences, resulting in different similarity ratings. Many real-world applications can be seen as C-STS tasks such as ranking a set of documents retrieved for the same query in Information Retrieval (IR) (Manning et al., 2008), comparing two answers for the same question in Question Answering (QA) (Risch et al., 2021), or measuring the strength of a semantic relation between two entities in Knowledge Graph Completion (KGC) (Yoo et al., 2024; Lin et al., 2024).

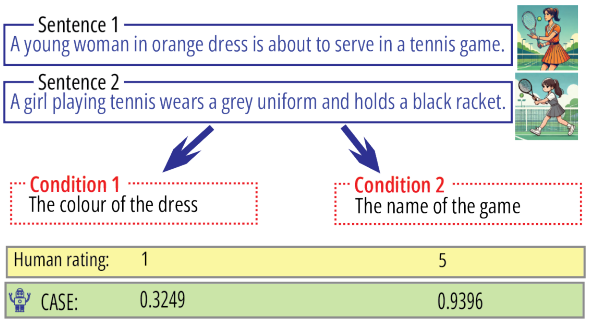


Figure 1: The two conditions focus on different information described in the two sentences. Human annotators rate the two sentences 1–5, indicating a high-level (5) of semantic textual similarity under condition 2 than condition 1 (1). Our proposed condition-aware sentence embedding (CASE) method reports similarity scores that are well-aligned with the human similarity ratings.

We propose **CASE**, a method for learning **C**ondition-**A**ware **S**entence **E**mbeddings, for a given input sentence considering another sentence. Specifically, CASE creates an embedding for the condition sentence using an LLM encoder with a

[1] Source code and pre-processed data for reproducing our findings are submitted to ARR and will be publicly released upon paper acceptance.

prompt that includes the target sentence (e.g. Sentences 1/2 in Figure 1), where the latter is not encoded explicitly in the embedding but influences the attention scores used during token pooling. Compared to Masked Language Models (MLMs) that have been used extensively in prior work on C-STS, LLMs contain billions of parameters and are typically trained on much larger datasets for a longer period of time. Consequently, LLM-based embedding models have consistently ranked at the top in leader-boards evaluating text embedding models (Muennighoff et al., 2022). Therefore, by leveraging LLM embeddings, CASE can benefit from the rich world knowledge contained in LLMs. However, the best way to use LLMs for C-STS remains elusive as reported by Lin et al. (2024) who showed that decoder-only LLMs often underperform MLM-based embeddings in C-STS benchmarks. Interestingly, we find that encoding the condition given the sentence often outperforms the reverse setting in C-STS benchmarks. Moreover, we find that subtracting the embedding of the condition in a post-processing step further improves the performance for multiple LLM encoders.

Compared to MLMs, LLM encoders have significantly larger (>1B) number of parameters, which makes it challenging to fine-tune them for C-STS even with parameter-efficient fine-tuning methods, without sacrificing their performance on other tasks. Moreover, LLM encoders such as `NV-embed-v2` (Lee et al., 2024) produce high-dimensional (4096) embeddings compared to MLM encoders such as RoBERTa-base (Liu et al., 2019) SimCSE (Gao et al., 2021) (768). High dimensional embeddings are problematic both due to their high storage requirements and the high inner-product computation cost. To address these issues, CASE trains a single feed-forward layer, which improves C-STS performance even in low-dimensional spaces. Further investigations into individual examples show that CASE increases the similarity between a sentence and the information emphasized by a given condition, while decreasing the same for information irrelevant to the condition as expected.

2 Related Work

Deshpande et al. (2023) proposed the C-STS task and created a dataset containing 18,908 instances where the semantic similarity between two sentences s_1 and s_2 is rated under two conditions c_{high} and c_{low} resulting in respectively high vs. low similarity between the two sentences. Moreover, they proposed cross-, bi- and tri-encoder baselines. Given a triplet (s_1, s_2, c) a cross-encoder considers all interactions among tokens in s_1, s_2 and c. Although the cross-encoder setting benefits from having access to both s_1 and s_2 at the same time, it is computationally costly due to the large number of token interactions required for longer inputs. Moreover, cross-encoders do not pre-compute conditional embeddings, which is problematic when computing all pairwise similarities between the sentences in a large set.

Bi-encoders overcome those limitations in cross-encoders by creating a condition-aware embedding for each sentence, and then compute C-STS efficiently by their inner-product. Tri-encoders separately encode s_1, s_2 and c, and then apply some late interactions between the condition's and each sentence's embeddings to compute C-STS. However, the late interaction mechanisms for tri-encoders remain complex report suboptimal performance.

Tu et al. (2024) found issues in the dataset created by Deshpande et al. (2023) such as ill-defined conditions and inconsistent similarity ratings for over half of the dataset. To address this, Tu et al. (2024) re-annotated the validation split from the original C-STS dataset. Moreover, they proposed a method to solve C-STS by first converting each condition into a question, and then using GPT-3.5 to extract the corresponding answers from the two sentences. Finally, the cosine similarity between the SimCSE embeddings for the two answers is computed. Although this QA formulation consistently improves the performance of all baseline encoders, it depends on multiple decoupled components such as converting conditions into questions, requiring a decoder LLM to extract the answer from each question, and using a separate encoder to compare the extracted answers, which increases the possibility of error propagation across components.

Zhang et al. (2025) further investigated the dataset and found issues such as problematic condition statements and inaccurate human annotations. They proposed an LLM-based two-step method to improve the dataset by first refining the condition statements and then re-annotating the ratings, achieving a larger scale (14176 instances) dataset of better quality, which we use for training in this paper.

Yoo et al. (2024) proposed Hyper-CL, a contrastively learnt hypernetwork (Ha et al., 2017) to

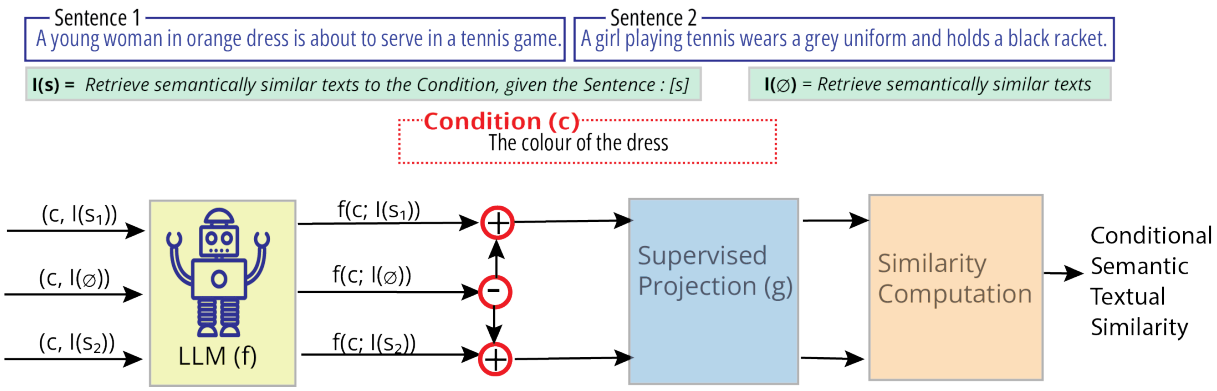


Figure 2: Overview of CASE. An LLM is prompted with $I(s)$ to obtain two separate embeddings $f(c; I(s_1))$ and $f(c; I(s_2))$ for the condition c for the two sentences s_1 and s_2. The unconditional embedding $f(c; I(\emptyset))$ is then computed using the prompt $I(\emptyset)$ and subtracted from each of those embeddings. Finally, the embeddings are projected to a lower-dimensional space using a learnt projection g and their cosine similarity is computed.

selectively project the embeddings of s_1 and s_2 according to c. Hyper-CL follows a tri-encoder setting where s_1, s_2 and c are first encoded separately using a sentence encoder. Next, a hypernetwork is trained to produce a linear transformation matrix conditioned on c. Finally, the embeddings of s_1 and s_2 are projected using this transformation matrix and their inner-product is computed. Hyper-CL improves the performance of tri-encoder models, but still under-performs bi-encoders for C-STS. Moreover, hypernetworks introduce an external parameter set that is three times larger than the SimCSE model used to encode each sentence, resulting in an excessively large memory space.

Lin et al. (2024) proposed a tri-encoder-based C-STS method where they used routers and heavy-light attention (Ainslie et al., 2023) to select the relevant tokens to a given condition. Specifically, they used the query vector of the [CLS] token of the condition to compute attention scores for the tokens in the sentence, which are subsequently used to compute a sentence embedding. Their method outperforms Hyper-CL for the tri-encoder setting. Liu et al. (2025) proposed a conditional contrastive learning method for C-STS, introducing a weighted contrastive losses with a sample augmentation strategy. Although it improved performance for both bi- and tri-encoders, the former outperforms the latter.

Li et al. (2024) proposed a cross-encoder approach which predicts C-STS scores, without creating conditional embeddings. Prior work on C-STS has shown cross-encoders to perform poorly despite having access to both sentences simultaneously. Li et al. (2024) showed that this is due to the irrelevant information in the two sentences to the condition and proposed a token re-weighting strategy, inspired by object detection in computer vision (Shi et al., 2023; Jaegle et al., 2021). Concretely, they compute two cross-attention matrices between (s_1, s_2) and c, which are subsequently used to compute the correlations for the sentence or condition tokens. Although their method improves the performance of cross-encoder-based C-STS measurement, it still underperforms bi-encoders.

Li and Li (2023) proposed BeLLM, which uses backward dependency to enhance LLMs by learning embeddings from uni- to bi-directional attention layers. They fine-tuned LLaMA2-7B model with their method on the C-STS task, outperforming previously proposed fine-tuned models. They designed a prompt template for this task: "*Given the context [condition], summarize the sentence [sentence] in one word:*". They then extracted embeddings from the hidden states of the generated output and used these embeddings to measure sentence similarity. Their results showed that larger model sizes and backward dependencies contribute to the fine-tuning effects. In contrast to our CASE, which uses LLM encoders, they use decoder-only LLMs. Moreover, we do not require token generation and therefore not restricted to single-word summarisation. Additionally, we find the reversed prompt structure ("*given [sentence], retrieve [condition]*") to perform better.

3 Condition-Aware Sentence Embeddings

An overview of our proposed method is shown in Figure 2, which consists of two-steps. In the first step (§3.1), we create two separate embeddings for the condition considering each of the two sentences,

one at a time, in the instruction prompt shown to an LLM. Note that it is the condition that is being encoded and the tokens in the sentence (similar to all other tokens in the instruction) are simply modifying the attention scores computed for the tokens in the condition. Intuitively, it can be seen as each sentence *filling* some missing information asked in the condition. Embeddings obtained from LLMs are not necessarily aligned to the C-STS task. Therefore, in the second step (§3.2), we learn a projection layer using the training split from the C-STS dataset. Finally, the C-STS between two sentences is computed as the cosine similarity between the corresponding projected embeddings.

3.1 Extracting Embeddings from LLMs

Given an LLM-based encoder, f, we create a d-dimensional embedding $f(c; I(s)) \in \mathbb{R}^d$ for a condition c, given the sentence s. Here, I is an instruction template that takes c as an argument. We use the following prompt template as $I(s)$ — *Retrieve semantically similar texts to the Condition, given the Sentence : [SENTENCE]*, where we substitute s in the placeholder [SENTENCE]. Next, we provide c as the input text to be encoded by the LLM following the instruction $I(s)$. Finally, the token embeddings of c are aggregated according to one of the pooling methods described previously to create $f(c; I(s))$. Although our focus here is to create condition-aware embeddings, it is noteworthy that we can also obtain *unconditional* embeddings for a text by dropping the condition in the above prompt. Specifically, we use the prompt $I(\emptyset)$ *Retrieve semantically similar texts to a given Sentence* for this purpose, and denote this unconditional embedding of c by $f(c; I(\emptyset))$. As we see later in our experiments, by subtracting $f(c; I(\emptyset))$ from $f(c; I(s))$, we can reduce the effect of tokens in the condition that are irrelevant to the sentence, thereby consistently improving accuracy of the condition-aware embeddings. This first step is fully unsupervised and a zero-shot prompt template is used as I.

Recall that both s and c are text strings, and it is possible to swap the sentence and condition in the above formulation to obtain an embedding $f(s; I(c))$ for the sentence, given the condition. However, as shown later in our experiments (§4.2), comparing the embedding for c created under s_1 and s_2 results in better performance on the C-STS benchmark for all LLM encoders (therefore, we keep the best-performing $f(c; I(s))$ in Figure 2). This is because s_1 and s_2 contain many irrelevant information to c, which will affect the cosine similarity computed between $f(s_1; I(c))$ and $f(s_2; I(c))$. On the other hand, the cosine similarity between $f(c; I(s_1))$ and $f(c; I(s_2))$ is a more accurate estimate of the C-STS between s_1 and s_2 under c because it is purely based on the meaning alterations to c under s_1 and s_2 separately.

3.2 Supervised Projection Learning

The LLM embeddings computed in §3.1 has two main drawbacks. First, although LLMs are typically trained on massive text collections and instruction-tuned for diverse tasks (Muennighoff et al., 2022), their performance on C-STS tasks have been poor (Lin et al., 2024). As seen from our condition-aware prompt template, an LLM must be able to separately handle a variable condition statement and a fixed instruction. This setup is different from most tasks on which LLMs are typically trained on, where the instruction remains fixed across all inputs. Second, relative to MLMs-based sentence embeddings, LLMs produce much higher dimensional embeddings, which can be problematic due to their memory requirements (especially when operating on a limited GPU memory) and the computational cost involved in inner-product computations. In tasks such as dense retrieval, we must compare millions of documents against a query to find the nearest neighbours under strict latency requirements, and low-dimensional embeddings are preferable.

To address the above-mentioned drawbacks, we propose a supervised projection learning method. Specifically, we freeze the model parameters of the LLM and use a Feed Forward Network (FFN) layer that takes $f(c; I(s))$ as the input and returns a k-dimensional ($k \leq d$) embedding $g(f(c; I(s); \theta)$, where θ denotes the learnable parameters of the FFN. Finally, we define $\text{CASE}(s, c)$ as the projection of the offset between the conditional and the unconditional embeddings of c under s, given by,

$$\text{CASE}(s, c) = g(f(c; I(s)) - f(c; I(\emptyset)); \theta). \quad (1)$$

We use the human-annotated similarity ratings r in the C-STS train instances $\mathcal{D}$ to learn θ. Specifically, we minimise the squared error between the human ratings and the cosine similarity computed using the corresponding CASE as given by (2).

$$\sum_{(s_1, s_2, c, r) \in \mathcal{D}} (\cos(\text{CASE}(s_1, c), \text{CASE}(s_2, c)) - r)^2 \quad (2)$$

Here, cos denotes the cosine similarity between the projected embeddings. We use Adam optimiser (Kingma and Ba, 2014) to find the optimal θ that minimises the loss given by (2). Recall that only the FFN parameters are updated during this projection learning step, while keeping the parameters of the LLM fixed, which makes it extremely efficient. For example, it takes less than 5 minutes to learn this projection even for the largest (4096 dimensional) embedding spaces using the train split of the C-STS dataset. Using the learnt projection, we compute the C-STS between s_1 and s_2 under c as the cosine similarity between the embeddings $\text{CASE}(s_1, c)$ and $\text{CASE}(s_2, c)$.

4 Experiments and Results

To evaluate the effectiveness of the LLM-based and MLM-based sentence embeddings as described in §3, we use six sentence encoders, out of which three are LLM-based: *NV-Embed-v2* (4096 dimensional **NV**), *SFR-Embedding-Mistral* (4096 dimensional **SFR**), *gte-Qwen2-7B-instruct* (3584 dimensional **GTE**) and three are MLM-based: *Multilingual-E5-large-instruct* (1024 dimensional **E5**), *sup-simcse-roberta-large* (1024 dimensional **SimCSE_large**), and *sup-simcse-bert-base-uncased* (768 dimensional **SimCSE_base**). Further details provided in Appendix A.

We evaluate model performance on different pooling methods, prompt settings, and sentence constructions. Moreover, we evaluate the supervised methods of linear and non-linear FFNs. Both the linear and non-linear FFNs are Siamese bi-encoders with weight-sharing. As explained in §3.2, they take $f(c; I(s_1))$ and $f(c; I(s_2))$ as the input embeddings, and return the projected embeddings $\text{CASE}(s_1, c)$ and $\text{CASE}(s_2, c)$ as the outputs, which can be compared using a similarity metric such as the cosine similarity.

Linear FFN performs a linear transformation:

$$\mathbf{z} = \text{Dropout}\left(\mathbf{W}\boldsymbol{e}\right) \tag{3}$$

where $\mathbf{W} \in \mathbb{R}^{d' \times d}$ is the learned projection matrix. A dropout layer is applied to reduce any overfitting (Hinton et al., 2012).

Our non-linear FFN is a two-layer FFN:

$$\mathbf{h} = \text{Dropout}\left(\text{ReLU}\left(\mathbf{W}_1\boldsymbol{e}\right)\right) \tag{4}$$
$$\mathbf{z} = \text{Dropout}\left(\text{ReLU}\left(\mathbf{W}_2\boldsymbol{h}\right)\right) \tag{5}$$

Hyperparameters are tuned on a held-out validation set. We select a learning rate of 10^{-3} and a batch size of 512. The dropout rate is set to 20% for the linear FFN and 15% for non-linear FFN.

Model	sent/cond?	Spear.
NV	sent - c	16.98
	sent	22.07
	cond - c	**37.61**
	cond	33.85
SFR	sent - c	19.54
	sent	11.89
	cond - c	**20.44**
	cond	18.32
GTE	sent - c	7.16
	sent	7.16
	cond - c	**20.40**
	cond	16.58
E5	sent - c	11.08
	sent	3.77
	cond - c	**15.37**
	cond	6.07
SimCSE_large	$\text{CONC}(s+c)$	5.59
	$\text{CONC}(c+s)$	4.00
	$\text{CONC}(s+c)$ - c	8.32
	$\text{CONC}(c+s)$ - c	**8.58**
SimCSE_base	$\text{CONC}(s+c)$	4.37
	$\text{CONC}(c+s)$	1.25
	$\text{CONC}(s+c)$ - c	**7.05**
	$\text{CONC}(c+s)$ - c	6.00

Table 1: Spearman scores for different sentence embedding models and encoding settings.

To address annotation errors in the original C-STS dataset such as ambiguous and invalid conditions, Tu et al. (2024) re-annotated the original validation set. Zhang et al. (2025) further identified the issues on both condition statements and ratings. They improved the dataset by first revising the condition statements and then re-annotating the similarity ratings for both the training and validation sets. To conduct a more accurate and reliable evaluation, we use the most recent revised and cleaned datasets by Zhang et al. (2025). We split their validation set into a validation/test (70% vs. 30%) randomly. We use a single p3.24xl EC2 instance (8x V100 GPUs) for learning sentence embeddings, and a separate NVIDIA RTX A6000 GPU for supervised projection learning. Pytorch 2.0.1 with cuda 11.7 is used for FFN proejction. These settings are fixed across all experiments. For reducing 4096-dimensional LLM-based sentence embeddings to 512-dimensional, training a linear FFN for CASE takes approximately 1.5 minutes, whereas a non-linear FFN requires about 5 minutes (wall-clock time).

Model	I_{iso} (-c)	I_{iso}
NV	0.9611	0.9471
SFR	0.9463	0.9266
GTE	0.9490	0.9310
E5	0.8906	0.8368
SimCSE_base	0.9461	0.9297
SimCSE_large	0.9499	0.9406

Table 2: I_{iso} values for each model. The left column of I_{iso} (-c) lists values with the post-processing step of subtracting the embedding of condition c. I_{iso} values close to 1 indicate high isotropy.

Model	Non-linear FFN	Linear FFN
NV	**73.93**	72.90
SFR	**66.84**	56.22
GTE	**64.16**	55.88
E5	**65.07**	48.25
SimCSE_large	**63.64**	48.15
SimCSE_base	**61.52**	42.22

Table 3: Spearman correlation of embedding models with supervised methods at reduced dimensionality 512.

4.1 Evaluation Metrics

We evaluate the performance of a sentence encoder by the Spearman correlation coefficient between the cosine similarity scores computed using the embeddings produced by an encoder and the human similarity ratings on the test set.

4.2 C-STS Measurement

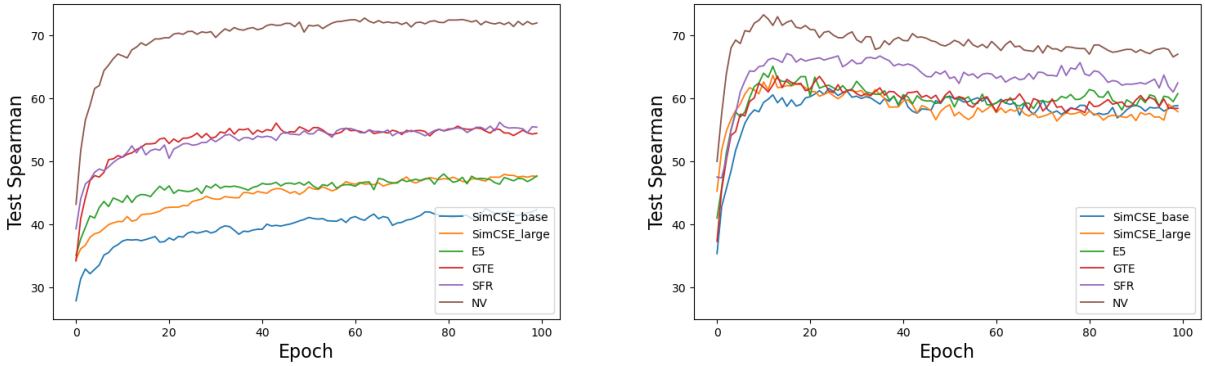
To generate CASE, we apply different ways to construct the prompt for LLM-based embeddings and to concatenate the condition and sentence for MLM-based embeddings. For LLM embeddings, we have two main settings: (a) $\text{cond} = f(c; I(s))$, where we encode the condition given the sentence, and (b) $\text{sent} = f(s; I(c))$, where we encode the sentence given the condition as explained in §3.1. For MLM embeddings, we evaluate the two settings: (a) $\text{CONC}(c + s)$, where we concatenate sentence after the condition, and (b) $\text{CONC}(s + c)$, where we concatenate condition after the sentence. For each setting, we evaluate the effect of subtracting the condition embedding, $\boldsymbol{c} = f(c; I(\emptyset))$.

The test performance for different settings and models are shown in Table 1. For LLM embeddings, **cond** consistently reports higher Spearman correlation than **sent**, suggesting that embedding the condition given the sentence is more effective for C-STS measurement than embedding the sentence given the condition. Moreover, we see that subtracting $\boldsymbol{c}$ further improves performance across all settings. The former approach reduces the noise due to the tokens in a sentence, which are irrelevant to the given condition. For MLM-based embeddings, subtracting $\boldsymbol{c}$ also improves performance. This operation removes the sentence-independent components of the condition, thereby isolating the contrastive semantics — the aspects of the condition that are actually altered by the sentence. As a result, CASE focuses on the semantic variation between the two sentences under the given condition, rather than the redundant information shared across all condition phrases.

Additionally, we discovered that subtracting $\boldsymbol{c}$ in a post-processing step improves isotropy of the embeddings (full details in Appendix B). We use the approximated isotropy metric I_{iso} (Durdy et al., 2023) as a numerically stable method to measure isotropy. I_{iso} is estimated by randomly sampling unit vectors on the hypersphere and computing the ratio of their minimum to maximum alignment with the embeddings across random directions. Table 2 shows that the post-processing step of subtracting the condition $\boldsymbol{c}$ gives higher I_{iso} values, indicating the improvement of isotropy. This makes subtle semantic differences between sentences under a given condition more distinguishable. This is inline with prior work reporting a positive correlation between isotropy and improved performance in embedding models (Rajaee and Pilehvar, 2022; Su et al., 2021). The effect of pooling method on performance is discussed in Appendix C, where we find the **latent** pooling in NV to perform best. Therefore, we use the $(\text{cond} - \boldsymbol{c})$ setting (which reports the best performance) for the six sentence encoders to conduct the subsequent experiments.

We show the training curves for our supervised FFNs in Figure 3. Overall, the non-linear FFNs achieve significantly higher Spearman correlation than the linear FFNs. Moreover, non-linear FFNs converge faster, typically reaching their peak performance within 20 epochs, after which the performance declines due to overfitting. The performance of the linear FFNs, gradually increases and eventually converges as the training progresses. NV consistently performs the best for both linear and non-linear FFNs.

We compare sentence encoders in Table 3 for a fixed (i.e. 512) dimensional projection from their original embeddings. NV obtains the highest Spearman coefficient across all sentence encoders and supervised methods. GTE and SFR also achieve high Spearman coefficients under the non-linear



(a) Test Spearman for linear FFN

(b) Test Spearman for non-linear FFN

Figure 3: Spearman correlation on test set for different models over training steps with dimensionality 512. The y-axes of both subfigures are aligned, facilitating a direct comparison of the Spearman coefficients across the two line charts, with the same colour for the same model. Best viewed in colour.

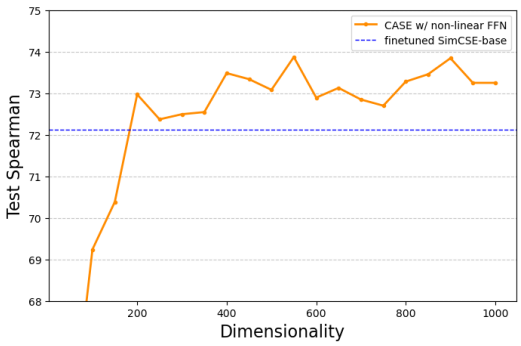


Figure 4: Spearman correlation coefficients of our CASE for NV embeddings on the test set over different dimensionalities. The horizontal dashed line represents the performance of fine-tuned SimCSE-base (768-dimensional).

Method	Spearman	dim	#params
Deshpande et al.			
w/ SimCSE-large	79.51	1024	354.3M
w/ SimCSE-base	72.12	768	124.1M
CASE			
w/ non-linear FFN	73.93	512	4.7M

Table 4: Comparison of Spearman correlation, embedding dimensionality, and trainable parameters of previously proposed methods and our CASE. #params denotes the number of trainable parameters.

FFNs. SimCSE_large and SimCSE_base have the lowest Spearman coefficients. Overall, we see that LLM-based embeddings perform better than the MLM-based embeddings on the C-STS task. Importantly, all embeddings are projected to the same lower dimensionality.

To study how the performance of the proposed CASE varies with the dimensionality of the supervised projection, we plot the Spearman correlation measured on the test split of the C-STS dataset in Figure 4 using NV embeddings. Because we are specifically interested in examining how CASE performs in the low-dimensional regime, which is attractive from a computational viewpoint, we consider dimensionalities in range [50, 1000]. Moreover, as a reference, we plot the performance of the previously proposed bi-encoder model by (Deshpande et al., 2023) for the SimCSE-base model, which is the smallest dimensional (768) encoder used in their work. From Figure 4 we see that for dimensionalities greater than 200, CASE consistently outperforms SimCSE-base bi-encoder. This result has practical implications because it shows that we can reduce the high dimensionality of LLMs-based (NV) embeddings from 4096 to 512 with an 8X compression, while preserving its high performance.

We compare CASE against the best bi-encoder models with fine-tuned SimCSE by Deshpande et al. (2023). As shown in Table 4, our proposed CASE with non-linear FFN achieves 73.9% Spearman correlation, outperforming fine-tuned SimCSE_base while using only 4.7M trainable parameters and a 512-dimensional embedding space. Compared with SimCSE_large, CASE maintains 93% of its performance, but requires 50% fewer dimensions and only 1.3% of its parameters. This highlights that CASE is a simple but an efficient method.

4.3 Case Study

To further illustrate how CASE captures condition-dependent semantics, Table 5 presents qualitative examples. Similarity scores are computed and

s1: Young woman in orange dress about to serve in tennis game, on blue court with green sides.	
s2: A girl playing tennis wears a gray uniform and holds her black racket behind her.	
cos(s1, s2) $0.5006 \rightarrow 0.9016$	
Condition 1: color of the dress	**Condition 2:** name of the game
Answer 1: orange **Answer 2:** gray **Rating:** 1	**Answer 1:** tennis **Answer 2:** tennis **Rating:** 5
cos(s1, s2; c1) $0.4757 \rightarrow 0.2522$	**cos(s1, s2; c2)** $0.6233 \rightarrow 0.9660$
cos(s1, orange; c1) $0.1986 \rightarrow 0.3551$	**cos(s1, tennis; c2)** $0.1135 \rightarrow 0.6448$
cos(s2, gray; c1) $0.0559 \rightarrow 0.6061$	**cos(s2, tennis; c2)** $0.0983 \rightarrow 0.6426$
s1: Two snow skiers with ski poles and snow skis, standing on top of a snow covered mountain with other skiers around them.	
s2: A skier stands alone at the top of a snowy slope with blue skies and mountains in the distance.	
cos(s1, s2) $0.6318 \rightarrow 0.7702$	
Condition 1: number of person	**Condition 2:** type of job
Answer 1: two **Answer 2:** one **Rating:** 1	**Answer 1:** skier **Answer 2:** skier **Rating:** 5
cos(s1, s2; c1) $0.6358 \rightarrow 0.2788$	**cos(s1, s2; c2)** $0.7502 \rightarrow 0.9539$
cos(s1, two; c1) $-0.0970 \rightarrow 0.7014$	**cos(s1, skier; c2)** $0.2488 \rightarrow 0.8016$
cos(s2, one; c1) $0.0227 \rightarrow 0.7953$	**cos(s2, skier; c2)** $0.3409 \rightarrow 0.8032$
s1: A bunch of people standing around at the beach with a kite in the air.	
s2: a beach scene with a beach chair decorated with the Canadian Flag and surfers walking by with their surfboards	
cos(s1, s2) $0.3988 \rightarrow 0.5350$	
Condition 1: type of hobby	**Condition 2:** type of location
Answer 1: kite flying **Answer 2:** surf **Rating:** 1	**Answer 1:** beach **Answer 2:** beach **Rating:** 5
cos(s1, s2; c1) $0.4386 \rightarrow 0.4886$	**cos(s1, s2; c2)** $0.4825 \rightarrow 0.8582$
cos(s1, kite flying; c1) $0.3457 \rightarrow 0.7717$	**cos(s1, beach; c2)** $0.2254 \rightarrow 0.7281$
cos(s2, surf; c1) $0.1034 \rightarrow 0.6665$	**cos(s2, beach; c2)** $0.1129 \rightarrow 0.6703$

Table 5: Example of similarity scores for two conditions applied to the same sentence pair, based on linear FFN with dimensionality 512 on NV-Embed-v2 (NV). The table shows how supervised FFN projection improves the CASE for C-STS task. $\cos(\cdot,\cdot)$ denotes cosine similarity. Answer 1 and Answer 2 refer to the corresponding answers for Sentence 1 and Sentence 2 under conditions. The predicted similarity scores before and after applying supervised FFN are listed on the left and right of the arrow. That is, the similarity score for original high-dimensional LLM-based embeddings is on the left of the arrow, while the similarity score for CASE is on the right.

compared for three sentence pairs for the original high-dimensional LLM-based embeddings and low-dimensional trained CASE. By comparing the similarity scores with and without the supervised projection, we can evaluate the ability of CASE to focus on the condition-related information. Compared to the unconditional similarity between the two sentences, conditional similarities scores computed using NV embeddings align well with the human ratings. For example, in the top row in Table 5, we see that the unconditional similarity between the two sentences reduces from 0.5006 to 0.4747 under c_1, while increasing to 0.6233 under c_2. Moreover, the conditional similarities are further appropriately amplified by CASE using the supervised projection (i.e. decreasing to 0.2522 under c_1, while increasing to 0.9660 under c_2). Specifically, CASE reduces the similarity between the two sentences under the lower-rated condition, while increasing the same under the high-rated condition.

We also treat the conditions as questions for sentences and extract the relevant information as answers to compare whether CASE can focus on the condition-related information. The overall similarity trend is consistent with the actual ratings. For all sentence and answer pairs, the similarity scores increase after supervised projection. This demonstrates that CASE can effectively capture the shift in conditional meaning between sentences under different conditions.

5 Conclusion

We propose CASE, which encodes the condition under sentence by subtracting the condition in a post-processing step when measuring C-STS. Our findings show that LLM embeddings consistently outperform MLM embeddings for C-STS. Moreover, we introduce an efficient supervised projection learning method to improve the performance in C-STS while projecting embeddings into lower-dimensional spaces for efficient computation.

6 Limitations

We use the original C-STS training set and the re-annotated validation set in this paper. The dataset, especially the training set, includes annotation er-

rors, such as invalid conditions and subjective human annotations. These disadvantages hurt the model's ability to capture the semantics and perform the C-STS task. This further introduces ambiguity and inaccuracy to CASE. Manually re-annotating the training set, which contains 11342 instances, is costly and is deferred to future work.

Our evaluations cover only English, which is a morphologically limited language. To the best of our knowledge, C-STS datasets have not been annotated for languages other than English, which has forced all prior work on C-STS to conduct experiments using only English data. However, the sentence encoders we used in our experiments support multiple languages. Therefore, we consider it to be an important future research direction to create multilingual datasets for C-STS and evaluate the effectiveness of our proposed method in multilingual settings.

There is a large number of sentence encoders (over 300 models as at October 2025 evaluated on Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023) leaderboard[2]). However, due to computational limitations, we had to select a subset covering the best performing (top ranked on MTEB at the time of writing) models for our evaluations. We will release our source code and the evaluation framework such that the research community can evaluate our proposed method with any sentence encoder for C-STS.

7 Ethical Concerns

We did not collect or annotate any datasets in this project. Instead, we use existing C-STS datasets annotated and made available by Deshpande et al. (2023) and Tu et al. (2024). To the best of our knowledge, no ethical issues have been raised regarding those datasets.

We use multiple pre-trained and publicly available MLM- and LLM-based sentence encoders (Kaneko and Bollegala, 2021). Both MLMs and LLMs are known to encode unfair social biases such as gender or racial biases. We have not evaluated how such social biases would be influenced by the CASE learning method proposed in this work. Therefore, we consider it would be important to measure the social biases in CASE created in this work before they are deployed in real-world applications.

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

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Matthew Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, I Lopez-Gazpio, M Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability. *SemEval*, pages 252–263.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. SemEval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 497–511, Stroudsburg, PA, USA. Association for Computational Linguistics.

Eneko Agirre, Daniel Matthew Cer, Mona T Diab, and Aitor Gonzalez-Agirre. 2012. SemEval-2012 task 6: A pilot on semantic Textual Similarity. *SemEval*, pages 385–393.

Joshua Ainslie, Tao Lei, Michiel de Jong, Santiago Ontanon, Siddhartha Brahma, Yury Zemlyanskiy, David Uthus, Mandy Guo, James Lee-Thorp, Yi Tay, Yun-Hsuan Sung, and Sumit Sanghai. 2023. CoLT5: Faster long-range transformers with conditional computation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5100, Stroudsburg, PA, USA. Association for Computational Linguistics.

Qian Chen, Wen Wang, Qinglin Zhang, Siqi Zheng, Chong Deng, Hai Yu, Jiaqing Liu, Yukun Ma, and Chong Zhang. 2023. Ditto: A simple and efficient approach to improve sentence embeddings. *Empir Method Nat Lang Process*, abs/2305.10786:5868–5875.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 681–691, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ameet Deshpande, Carlos E Jimenez, Howard Chen, Vishvak Murahari, Victoria Graf, Tanmay Rajpurohit, Ashwin Kalyan, Danqi Chen, and Karthik Narasimhan. 2023. CSTS: Conditional semantic textual similarity. *Empir Method Nat Lang Process*, pages 5669–5690.

Samantha Durdy, Michael W. Gaultois, Vladimir Gusev, Danushka Bollegala, and Matthew J. Rosseinsky. 2023. Metrics for quantifying isotropy in high dimensional unsupervised clustering tasks in a materials context. *Preprint*, arXiv:2305.16372.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. SimCSE: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on*

Empirical Methods in Natural Language Processing, pages 6894–6910, Stroudsburg, PA, USA. Association for Computational Linguistics.

David Ha, Andrew M Dai, and Quoc V Le. 2017. HyperNetworks. In *International Conference on Learning Representations*.

Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R. Salakhutdinov. 2012. Improving neural networks by preventing co-adaptation of feature detectors. *Preprint*, arXiv:1207.0580.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, Olivier J Henaff, Matthew Botvinick, Andrew Zisserman, Oriol Vinyals, and Joao Carreira. 2021. Perceiver IO: A general architecture for structured inputs & outputs. In *International Conference on Learning Representations*.

Masahiro Kaneko and D Bollegala. 2021. Unmasking the mask - evaluating social biases in masked language models. *National Conference on Artificial Intelligence*, pages 11954–11962.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *Int Conf Learn Represent*, abs/1412.6980.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024. NV-embed: Improved techniques for training LLMs as generalist embedding models. *arXiv [cs.CL]*.

Baixuan Li, Yunlong Fan, and Zhiqiang Gao. 2024. SEAVER: Attention reallocation for mitigating distractions in language models for conditional semantic textual similarity measurement. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 78–95, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xianming Li and Jing Li. 2023. Bellm: Backward dependency enhanced large language model for sentence embeddings. *arXiv preprint arXiv:2311.05296*.

Ziyong Lin, Quansen Wang, Zixia Jia, and Zilong Zheng. 2024. Varying sentence representations via condition-specified routers. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17390–17401, Stroudsburg, PA, USA. Association for Computational Linguistics.

Xinyue Liu, Zeyang Qin, Zeyu Wang, Wenxin Liang, Linlin Zong, and Bo Xu. 2025. Conditional semantic textual similarity via conditional contrastive learning. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 4548–4560.

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 [cs.CL]*.

Christopher D Manning, Prabhakar Raghavan, and Hinrich Schutze. 2008. *Introduction to Information Retrieval*. Cambridge University Press.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. MTEB: Massive text embedding benchmark. *arXiv [cs.CL]*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2023. Mteb: Massive text embedding benchmark. *Preprint*, arXiv:2210.07316.

Sara Rajaee and Mohammad Taher Pilehvar. 2022. An isotropy analysis in the multilingual bert embedding space. *Preprint*, arXiv:2110.04504.

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 3980–3990, Stroudsburg, PA, USA. Association for Computational Linguistics.

Julian Risch, Timo Möller, Julian Gutsch, and Malte Pietsch. 2021. Semantic answer similarity for evaluating question answering models. In *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pages 149–157, Stroudsburg, PA, USA. Association for Computational Linguistics.

Baifeng Shi, Trevor Darrell, and Xin Wang. 2023. Top-down visual attention from analysis by synthesis. *arXiv [cs.CV]*.

Jianlin Su, Jiarun Cao, Weijie Liu, and Yangyiwen Ou. 2021. Whitening sentence representations for better semantics and faster retrieval. *Preprint*, arXiv:2103.15316.

Jingxuan Tu, Keer Xu, Liulu Yue, Bingyang Ye, Kyeongmin Rim, and James Pustejovsky. 2024. Linguistically conditioned semantic textual similarity. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1161–1172, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jiahao Xu, Wei Shao, Lihui Chen, and Lemao Liu. 2023. SimCSE++: Improving contrastive learning for sentence embeddings from two perspectives. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 12028–12040, Stroudsburg, PA, USA. Association for Computational Linguistics.

Young Yoo, Jii Cha, Changhyeon Kim, and Taeuk Kim. 2024. Hyper-CL: Conditioning sentence representations with hypernetworks. In *Proceedings of the*

62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 700–711, Bangkok, Thailand. Association for Computational Linguistics.

Gaifan Zhang, Yi Zhou, and Danushka Bollegala. 2025. Annotating training data for conditional semantic textual similarity measurement using large language models. *arXiv preprint arXiv:2509.14399*.

Supplementary Materials

A Models

To evaluate the effectiveness of the LLM-based and MLM-based sentence embeddings as described in §3, we apply six sentence encoders, out of which three are LLM-based: *NV-Embed-v2* (4096 dimensional and uses latent pooling) (**NV**)[3], *SFR-Embedding-Mistral* (4096 dimensional and uses average pooling) (**SFR**)[4], *gte-Qwen2-7B-instruct* (3584 dimensional and uses last token pooling) (**GTE**)[5] and three are MLM-based: *Multilingual-E5-large-instruct* (1024 dimensional and uses average pooling) (**E5**)[6], *sup-simcse-roberta-large* (1024 dimensional and uses last token pooling) (**SimCSE_large**)[7], and *sup-simcse-bert-base-uncased* (786 dimensional and uses last token pooling) (**SimCSE_base**)[8].

B Isotropy for Embeddings

We first use the embedding-to-mean cosine similarity distribution to measure the isotropy of the embeddings. Given a set of embeddings $S = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n\}$, we first compute the mean embedding vector $\boldsymbol{\mu} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i$. Then, for each embedding $\boldsymbol{x}_i \in S$, we compute its cosine similarity with the mean vector μ, i.e., $\cos(\boldsymbol{x}_i, \boldsymbol{\mu}) = \frac{\boldsymbol{x}_i{}^\top \boldsymbol{\mu}}{||\boldsymbol{x}_i|| ||\boldsymbol{\mu}||}$. The distribution of these embedding-to-mean cosine similarities is then analysed to characterise the embedding space – a distribution sharply peaked near 1 indicates anisotropy, whereas a broader, more uniform distribution suggests a more isotropic geometry.

[3] https://huggingface.co/nvidia/NV-Embed-v2
[4] https://huggingface.co/Salesforce/SFR-Embedding-Mistral
[5] https://huggingface.co/Alibaba-NLP/gte-Qwen2-7B-instruct
[6] https://huggingface.co/intfloat/multilingual-e5-large-instruct
[7] https://huggingface.co/princeton-nlp/sup-simcse-roberta-large
[8] https://huggingface.co/princeton-nlp/sup-simcse-bert-base-uncased

Model	Embedding Type	Mean	Std
NV	cond - c	0.407	0.084
	cond	0.492	0.067
SFR	cond - c	0.537	0.055
	cond	0.708	0.021
GTE	cond - c	0.489	0.067
	cond	0.696	0.036
E5	cond - c	0.542	0.056
	cond	0.897	0.010
SimCSE_base	CONC$(c+s)$ - c	0.254	0.095
	CONC$(c+s)$	0.347	0.088
SimCSE_large	CONC$(c+s)$ - c	0.248	0.110
	CONC$(c+s)$	0.333	0.085

Table 6: Cosine similarity to mean vector: comparing mean and standard deviation of two embedding types across three LLM-based and three MLM-based models.

From Table 6, Figure 5, and Figure 6, we see that the embeddings after subtracting c have a lower mean cosine similarity to the mean vector and a higher standard deviation, indicating that they are more spread out in the embedding space. In contrast, the embeddings without subtracting c are more clustered around a central direction (higher mean, lower standard deviation), reflecting anisotropy, a tendency for vectors to concentrate in a narrow region. Therefore, embeddings after subtracting c tend to be more isotropic, indicating better distributional diversity.

We use an approximation method to compute the isotropy (Durdy et al., 2023). Given a normalized embedding matrix $E \in \mathbb{R}^{n \times d}$ where each row represents a unit vector, instead of computing the eigenvectors of the covariance matrix, we randomly sample k unit vectors $\boldsymbol{u}_1, \boldsymbol{u}_2, ..., \boldsymbol{u}_k$ from the unit hypersphere $\mathbb{S}^{d-1}$. For each sampled direction $\boldsymbol{u}_i$, we compute the function $F(\boldsymbol{u}_i) = \sum_{j=1}^{n} \exp(\boldsymbol{e}_j{}^\top \boldsymbol{u}_i)$, where $\boldsymbol{e}_j$ represents the j-th embedding vector. The estimated isotropy is then defined as the ratio between the minimum and maximum values of F, given by (6).

$$I_{\text{iso}} \approx \frac{\min_{i \in 1,\ldots,k} F(u_i)}{\max_{i \in 1,\ldots,k} F(u_i)} \tag{6}$$

If I_{iso} is close to 1, it means that the embedding space shows high isotropy where vectors are uniformly distributed across all directions in the high-dimensional space instead of clustering in some dominant directions. Conversely, when I_{iso} approaches 0, it means significant anisotropy in the

Model	I_{iso} (-c)	I_{iso}
NV	0.9611	0.9471
SFR	0.9463	0.9266
GTE	0.9490	0.9310
E5	0.8906	0.8368
SimCSE_base	0.9461	0.9297
SimCSE_large	0.9499	0.9406

Table 7: Isotropy values I_{iso} for each model. The left column of I_{iso} (-c) lists values with post-processing step of subtracting the condition c.

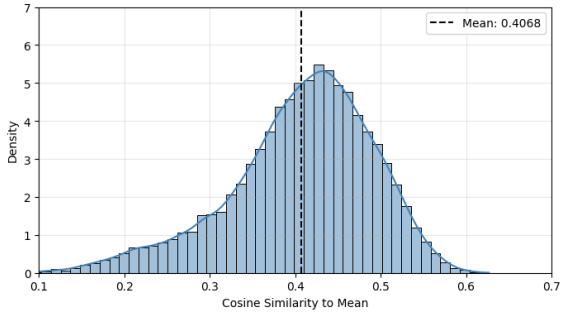
embedding space. We use $k = 1000$ random directions to compute the approximation of isotropy. Table 7 shows that the post-processing step of subtracting the condition c gives higher I_{iso} values, indicating the improvement of isotropy.

C Full Results for all Models

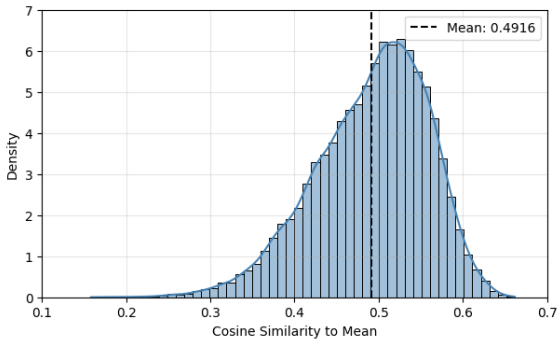
Each LLM encoder has its own preferred pooling method, recommended by the original authors of those models. From Table 8 we see that the performance varies significantly depending on the pooling method being used, while the latent attention pooling used in **NV** reporting the best results. Note that NV does not support **last** or **average** pooling, while **latent** pooling is not supported by the other models.

Model	Pooling	sent/cond?	Spear.
NV	latent	sent - c	16.98
	latent	sent	22.07
	latent	**cond - c**	**37.61**
	latent	cond	33.85
SFR	last	sent - c	11.88
	last	sent	-0.18
	last	cond - c	19.28
	last	cond	13.00
	average	sent - c	19.54
	average	sent	11.89
	average	**cond - c**	**20.44**
	average	cond	18.32
GTE	last	sent - c	7.16
	last	sent	7.16
	last	**cond - c**	**20.40**
	last	cond	16.58
	average	sent - c	13.01
	average	sent	11.34
	average	cond - c	17.87
	average	cond	18.42
E5	average	sent - c	11.08
	average	sent	3.77
	average	**cond - c**	**15.37**
	average	cond	6.07
	last	sent - c	9.28
	last	sent	0.49
	last	cond - c	8.90
	last	cond	3.05
SimCSE_large		CONC$(s+c)$	5.59
		CONC$(c+s)$	4.00
		CONC$(s+c)$ - c	8.32
		CONC$(c+s)$ - c	**8.58**
SimCSE_base		CONC$(s+c)$	4.37
		CONC$(c+s)$	1.25
		CONC$(s+c)$ - c	**7.05**
		CONC$(c+s)$ - c	6.00

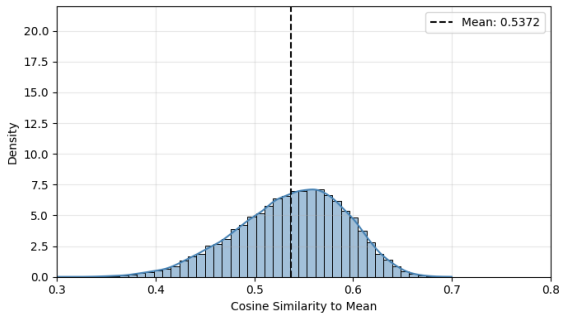
Table 8: Spearman scores for sentence embedding models. The original dimensionality of each model is indicated in parentheses. **latent** denotes latent attention pooling for NV, whereas **last** and **average** correspond to last token pooling and average pooling, respectively.



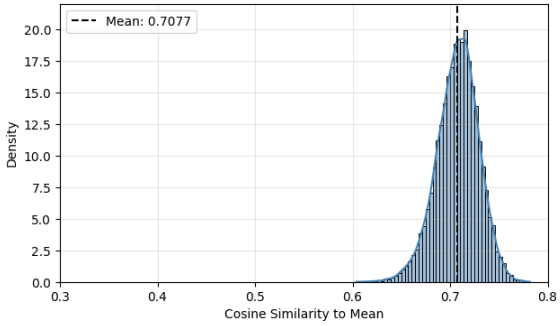
(a) **NV**: cond − c



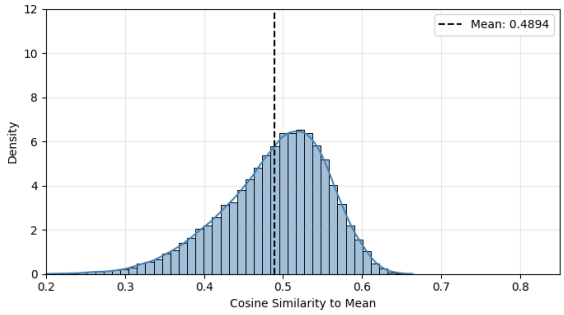
(b) **NV**: cond



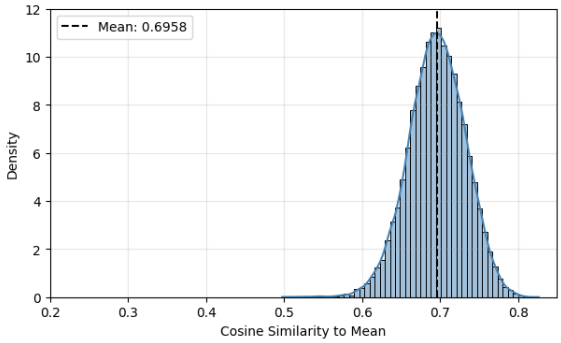
(c) **SFR**: cond − c



(d) **SFR**: cond

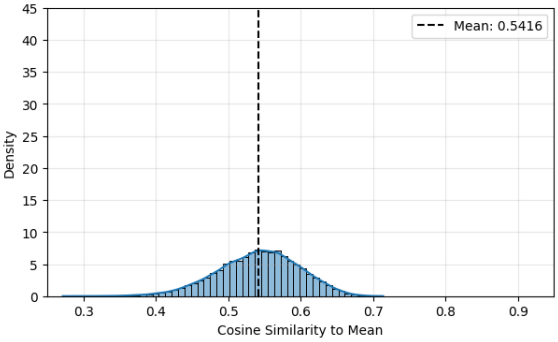


(e) **GTE**: cond − c

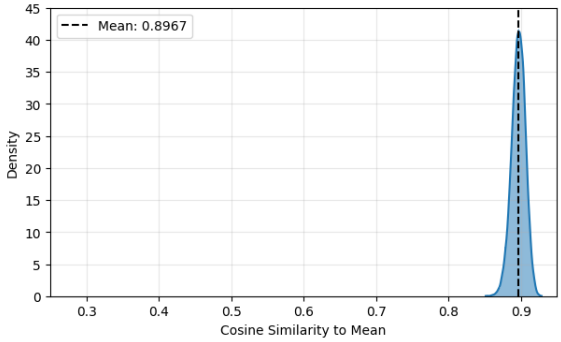


(f) **GTE**: cond

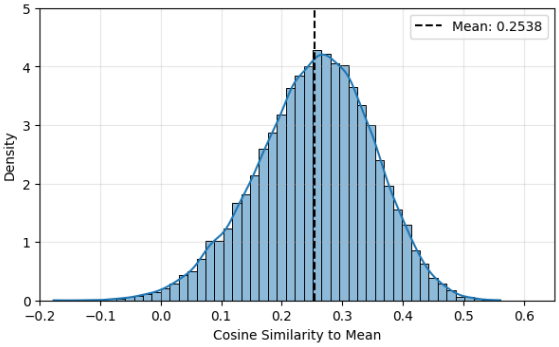
Figure 5: Embeddings-to-mean cosine similarity distributions across three LLM-based models. Each row compares cond − c and cond representations.



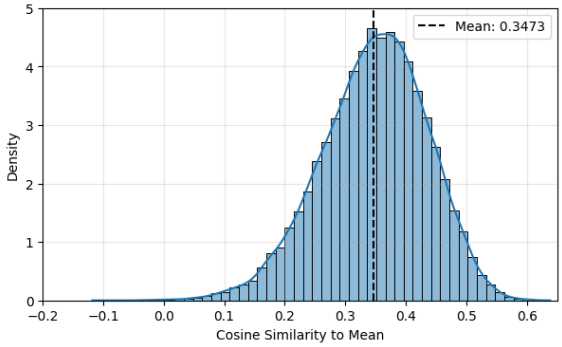
(a) **E5**: cond − c



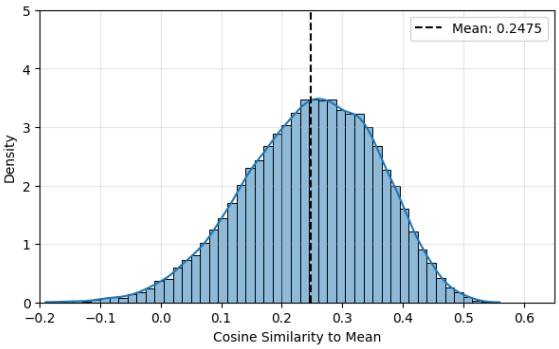
(b) **E5**: cond



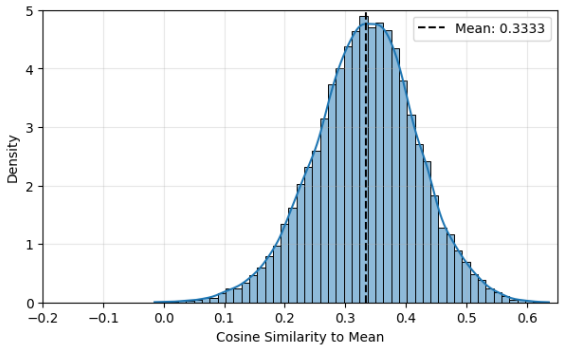
(c) **SimCSE_base**: $\mathrm{CONC}(c+s)$ - c



(d) **SimCSE_base**: $\mathrm{CONC}(c+s)$



(e) **SimCSE_large**: $\mathrm{CONC}(c+s)$ - c



(f) **SimCSE_large**: $\mathrm{CONC}(c+s)$

Figure 6: Embeddings-to-mean cosine similarity distributions across three MLM-based models. Each row compares two embedding types.