GenEOL: Harnessing the Generative Power of LLMs for Training-Free Sentence Embeddings

Anonymous ACL submission

Abstract

Training-free embedding methods directly leverage pretrained large language models (LLMs) to embed text, bypassing the costly and complex procedure of contrastive learning. Previous training-free embedding methods have mainly focused on optimizing embedding prompts and have overlooked the benefits of utilizing the generative abilities of LLMs. We propose a novel method, GenEOL, which uses LLMs to generate diverse transformations of a sentence that preserve its meaning, and aggregates the resulting embeddings of these transformations to enhance the overall sentence embedding. GenEOL significantly outperforms the existing training-free embedding methods by an average of 2.85 points across several LLMs on the sentence semantic text similarity (STS) benchmark. Our analysis shows that GenEOL stabilizes representation quality across LLM layers and is robust to perturbations of embedding prompts. GenEOL also achieves notable gains on multiple clustering, reranking and pair-classification tasks from the MTEB benchmark.

1 Introduction

While LLMs are very good at generating text, the embeddings obtained from their activations are often not suitable for downstream tasks such as computing sentence similarity or clustering. Contrastive learning (CL) is typically used to further finetune LLMs to produce better embeddings. Traditional contrastive learning approaches used human-annotated data for training embeddings (Gao et al., 2021), while recent methods leverage LLMs to generate contrastive data (Wang et al., 2023; Zhang et al., 2023a; Thirukovalluru et al., 2024b). However, curating high-quality CL training data is both costly and time-consuming, and CL further requires large batch sizes and extensive computational resources for each round of training (Wang et al., 2023; Muennighoff et al., 2024). As newer language models are continuously developed and released, it is important to consider training-free, inference-time text embedding methods, which may provide a more efficient and adaptable alternative to training-intensive techniques.

Jiang et al. (2023) first explored this task for sentences by proposing prompts like *This sentence: "[TEXT]" means in one word:"* and used the hidden layer representation of the last token from an LLM to represent *[TEXT]*. Zhang et al. (2024) enhanced the approach by incorporating chain-of-thought and knowledge reasoning. Similarly, Lei et al. (2024) utilized multiple diverse prompts to capture different sentence aspects, averaging their embeddings for improved results. Concurrently, Springer et al. (2024) repeated the original sentence to facilitate bidirectional attention in embedding sentences.

While these methods have significantly advanced training-free embeddings, they have not fully exploited the generative capabilities of LLMs to improve embedding quality. Recently, multiple techniques have begun scaling inference-time compute to enhance the reasoning and generative abilities of LLMs (Brown et al., 2024; Liang et al., 2024); we adopt a similar approach for embeddings in this work. Our approach offers a method to harness the generative capabilities of large language models (LLMs), to enhance sentence embeddings. It can also work with black-box language models. Specifically, we prompt an LLM to generate m diverse sentence variations that maintain the original meaning, which are then aggregated to produce more refined embeddings.

We specifically focus on the semantic text similarity (STS) benchmark for sentence embeddings (Conneau and Kiela, 2018). Our key contributions are as follows: 1. We show that diverse meaning retaining transformations are helpful in improving training free embeddings. 2. The proposed method, GenEOL, at a higher capacity (m=32), significantly outperforms the previous best training-

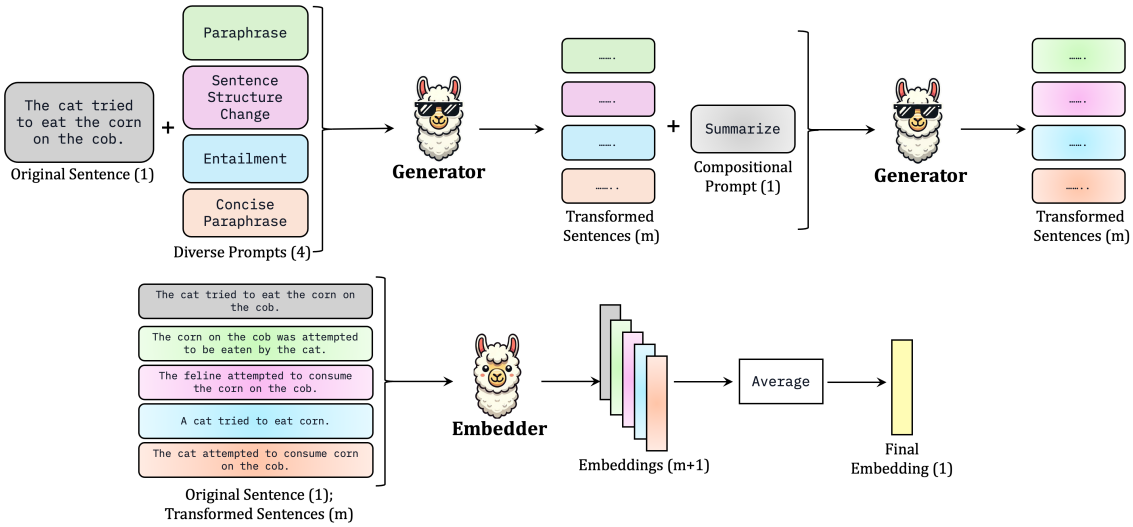


Figure 1: GenEOL methodology outlined. Step 1: *Generator* ($\mathcal{L}_{\mathbf{IT}}$) creates a set of transformed sentences, each conveying the same core meaning as the original sentence. Step 2: Original sentence, transformed sentences are embedded using *Embedder* ($\mathcal{L}_{\mathbf{PT}}$) and averaged to produce the final embedding. (.) is the count of each element.

free method by (3.88, 1.83, 2.83) points on average with (Mistral0.1-7B, Llama2-7B, Llama3-8B) respectively. GenEOL with even a small number of transformed sentences (m=2) surpasses all baselines. 3. GenEOL stabilizes representational quality across different layers of LLMs. GenEOL is robust to changes in embedding prompts. 4. GenEOL also performs better than baselines across multiple diverse MTEB tasks (Muennighoff et al., 2022).

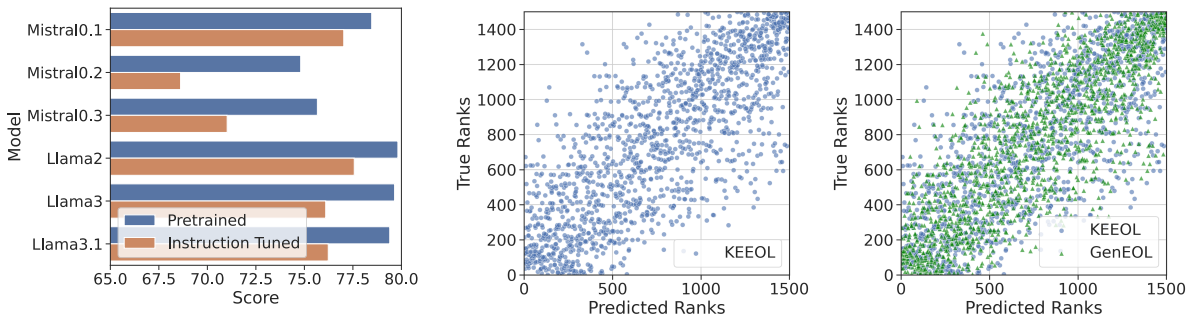
2 Background and Related Work

This section covers contrastive learning (CL) and generation-based methods utilizing CL, followed by an overview of training-free approaches.

CL Training: Contrastive training employs *(anchor, positive, negative)* data, where the positive is semantically similar to the anchor and the negative is dissimilar. InfoNCE loss (Gao et al., 2021) draws the positives' embeddings closer to that of the anchor while distancing the negatives. SimCSE (Gao et al., 2021) applies this loss using human-annotated data to train sentence embedding models.

Generating data for CL Training: E5 (Wang et al., 2023) used ChatGPT to generate a huge CL corpus of related positives and unrelated negatives across multiple tasks. Gecko (Lee et al., 2024) used an LLM to generate queries and relabel positives, negatives from existing data corpus. LLM2Vec (BehnamGhader et al., 2024) used representations generated from different dropout masks as positives with a positive only CL loss. For sentences specifically, SynCSE (Zhang et al., 2023a) defined specific transformations with ChatGPT to develop contrastive positives, negatives for sentence embedding training. Inspired by compositional transformations in CL data for computer vision (CV), SumCSE (Thirukovalluru et al., 2024b) further improved these transformations by using *'summary'* operations (akin to cropping in CV). Our method, GenEOL, takes inspiration from SynCSE, SumCSE in generating transformations (with same core meaning) to improve embedding quality.

Training Free Methods: Echo (Springer et al., 2024) showed that repeating the original sentence and using the hiddden representations of the later as the embedding of the sentence can improve performance. PromptEOL (Jiang et al., 2023) showed that generative LLMs when used with the prompt (*This sentence: "[TEXT]" means in one word:"*) and using the hidden representation of the last token i.e. *"* is very effective at embedding the text. Further improving this prompt Zhang et al. (2024) proposed two advanced prompts - 1. Pretended Chain of Thought prompt (PCoTEOL) - (*After thinking step by step , this sentence : "[X]" means in one word:"*) and 2. Knowledge Enhancement prompt (KEEOL) - (*The essence of a sentence is often captured by its main subjects and actions.... "[X]" means in one word:"*). MetaEOL (Lei et al., 2024) utilized eight diverse prompts from four categories—Text Classification, Sentiment Analysis, Paraphrase Identification, and Information Extraction—to aggregate multiple perspectives into a single final embedding. These methods also discuss better ways to extract text embeddings by using LLM penultimate layers. However, they do not fully leverage the generative power of LLMs for embedding sentences.



(a) (b) (c)

Figure 2: (a) Comparison of *Pretrained* vs the *Instruction Tuned* Models on STSB validation set. Pretrained models are always better in sentence embeddings. (b) Scatter Plot for score ranks on the STSB validation set using EOL based prompt. (c) Scatter Plot for score ranks on STSB validation set with `GenEOL`. The points are more concentrated along the diagonal compared to `KEEOL` (Note the blue dots away from the diagonal in (c)).

3 Motivation and Methodology

3.1 Motivation

This section outlines the STS sentence similarity task, presents key empirical observations that motivate our approach, and finally elaborates the proposed `GenEOL` method.

Task Definition: Given a set of sentence pairs, $\{...(x_{i1}, x_{i2})...\}$, the goal is to generate embeddings for each sentence such that, when ranked by the cosine similarity of their embeddings, the ranking aligns with a provided reference ranking. Performance is evaluated using Spearman's rank correlation between the predicted ranks and the true ranks. The task assumes access to a pretrained LLM $\mathcal{L}_{PT}$ and an instruction-tuned LLM $\mathcal{L}_{IT}$.

Observation 1: Instruction tuned LLMs are worse than Pretrained LLMs at Embeddings We first analyze in Fig. 2a, the performance (spearman rank correlation) of instruction-tuned LLMs versus pretrained models on the STSB validation set using the `KEEOL` prompt (Zhang et al., 2024) for sentence encoding. Results indicate that pretrained models consistently outperform instruction-tuned models in embeddings. Additionally, when applying an instruction version of the knowledge EOL prompt (with `[INST]` tokens) to instruction-tuned models, performance decreased further. This may explain why recent studies Zhang et al. (2024) and Lei et al. (2024) benchmark non-CL training-free methods primarily using pretrained models. On the other hand, instruction tuned models are generally vastly superior at generating texts suited to various prompts. This raises the question – how can we utilize the generative ability of an instruction-tuned model to enhance the embeddings obtained from the pretrained model?

Observation 2: LLM embedding scores are coarsely aligned with True scores. To visualize the rank correlation discussed above, we present a scatter plot comparing the true and predicted ranks for the `Mistral0.1-7B` embedding model in Fig. 2b. An ideal such scatterplot would have all elements aligned along the diagonal. Despite the notable amount of spread in Fig. 2b, predicted scores are still coarsely aligned with the true scores.

To reduce the spread around the diagonal, let's consider the following exercise. Let (x_{i1}, x_{i2}) be the i-th datapoint. Let τ_i be the true score and ρ_i be the predicted similarity score of this datapoint. Considering ρ_i to be a random variable, the coarse alignment trend from Fig 2b gives a sense of an error being present in ρ_i. Specifically, $\rho_i = \tau_i + \epsilon_i$, where ϵ_i is the error term with a mean μ_i and variance σ_i^2. From a statistics standpoint, the primary task now is to the reduce this error to improve performance. μ_i here is the inherent bias and would require further tuning the model to reduce it. To reduce the variance σ_i^2, a simple trick now would be to average k independent estimates of ρ_i i.e. $\{\rho_i^1, .., \rho_i^k\}$. It is known that the variance of the mean of k independent random variables decreases by a factor of k and hence reduces the error ϵ_i.

As we've seen in Observation 1, instruction tuned models although bad embedding models are very good at generating text of preferred format. Hence, we hypothesize that instruction tuned models can be prompted appropriately to transform datapoint (x_{i1}, x_{i2}) to $\{(x_{i1}^0, x_{i2}^0), (x_{i1}^1, x_{i2}^1), .., (x_{i1}^k, x_{i2}^k)\}$ with each pair containing the exact same meaning i.e. having the exact same τ_i. While these pairs may not be entirely independent, it would still contribute to reducing variance. Let's say we transform each

sentence m times and use their mean embedding to represent the sentence x_{i1}, i.e., $\sum_{j=0}^{m} \frac{h(x_{i1}^{(j)})}{m}$. This would result in $k =\sim m^2$ pairs in estimating $\rho_i^{mean} = \left\langle \left(\sum_{j=0}^{m} \frac{h(x_{i1}^{(j)})}{m} \right), \left(\sum_{j=0}^{m} \frac{h(x_{i2}^{(j)})}{m} \right) \right\rangle$ where $< .,. >$ is the dot product. Therefore, smaller values of m might still work pretty well. Fig. 2c shows the scatter plot with GenEOL($m = 8$). As shown, a notable number of blue dots that were initially spread far from the diagonal, moved closer to the diagonal with GenEOL (green triangles).

We now curate appropriate prompts that can realise the m transformed sentences while preserving meaning. We take inspiration from CL training data generation methods - SumCSE, SynCSE in doing so.

3.2 Methodology

Figure 1 illustrates the methodology for the proposed approach. Our method, GenEOL uses the LLMs in two distinct roles 1. **Generator (uses $\mathcal{L}_{\mathbf{IT}}$)**; 2. **Embedder (uses $\mathcal{L}_{\mathbf{PT}}$)**. Although our motivation necessitates independent transformations of x_i, the method to achieve this remains uncertain. Therefore, we employ the generator to facilitate diverse transformations of x_i. The transformations are accomplished by applying suitable prompts to the input sentence x_i and utilizing LLM $\mathcal{L}_{IT}$ to generate the modified sentence.

3.2.1 Diverse Transformations

Transformation 0 ($\mathcal{T}_0$): Original sentence, x_i, is the only transformation that retains all aspects and meaning of the sentence.

Transformation 1 ($\mathcal{T}_1$): Syntax of sentences has been shown to confuse sentence embeddings (Zhang et al., 2023b). Hence, $\mathcal{T}_1$ is a sentence structure changing transformation.

Transformation 2 ($\mathcal{T}_2$): Removing non essential details like adverbs shouldn't change the core meaning of the sentence. Hence, following Zhang et al. (2023a), we use concise sentence transformation as $\mathcal{T}_2$.

Transformation 3 ($\mathcal{T}_3$): Entailment is another transformation which has low semantic overlap while retaining the core meaning of a sentence (Gao et al., 2021). This becomes $\mathcal{T}_3$.

Transformation 4 ($\mathcal{T}_4$): A regular paraphrasing that can retain the meaning is used as $\mathcal{T}_4$ (Zhang et al., 2023a; Thirukovalluru et al., 2024b).

Few shot prompts used for these transformations are detailed in §A.4. An example of transformed sentences with these prompts is shown in Table 9 (§A). These prompts and demonstrations emphasize that the rephrased sentence must convey the same meaning as the original sentence. To evaluate the effectiveness of these diverse transformations, we analyze their impact on the STSB validation set. Table 1 shows that using diverse transformations to aggregate a sentence embedding is significantly more effective than just using the original sentence.

Transformation	STSB
Baseline	78.47
Diverse Transformations	82.21
Diverse Transformations + Composition Transformation	82.68

Table 1: Diverse transformations significantly improve performance on the STSB validation set. Compositional summary transformation yields notable gains.

3.2.2 Further Increasing Diversity (Optional)

SumCSE (Thirukovalluru et al., 2024b) showed that compositional summary transformations (a second summary transformation over the first transformation) is very effective at creating transformed sentences far from the original sentence. Compositional transformations retain the original meaning of the sentence while more significantly altering its lexical form compared to a single transformation. As shown in Table 1, a compositional summary transformation (§A.4) (emphasizes for meaning preservation in contrast to SumCSE prompt) shows benefits. Thus, we optionally incorporate the compositional summary transformation in our approach. The proposed framework, GenEOL, also allows for use of other diverse transformations or sampling multiple diverse sentences from these proposed transformations to improve sentence embeddings.

3.2.3 Final Embedding

The transformed sentences are individually embedded using the embedder, LLM $\mathcal{L}_{PT}$ following previous work (Zhang et al., 2024). This mean embedding becomes the final embedding of the sentence.

4 Experiments

We evaluate GenEOL and other baselines on 7 semantic text similarity tasks - STS12, STS13, STS14, STS16, STS16, STSB, SICK-R (Gao et al., 2021). Spearman rank correlation (cosine similarity) is the main metric (Muennighoff et al., 2022). Training sets of the STS tasks are not used. We additionally asses GenEOL on 10 MTEB tasks across 4 categories (Classification, Clustering, Reranking

Method	Gennerator	m	STS12	STS13	STS14	STS15	STS16	STSB	Sick-R	Avg.
Embedder: Mistral0.1-7B										
Token Avg.†	-	-	41.13	54.08	43.99	56.94	53.80	42.99	52.32	49.32
Echo†			58.43	78.53	68.42	78.82	77.52	73.85	71.95	72.5
PromptEOL†			63.08	78.58	69.40	77.92	79.01	75.77	69.47	73.32
PCoTEOL†			66.45	82.04	72.24	77.93	79.36	76.66	71.06	75.11
KEEOL†			66.33	81.52	71.73	77.53	77.99	74.09	74.02	73.89
MetaEOL†			64.05	82.35	71.57	81.36	79.98	78.29	75.13	76.09
GenEOL	Mistral0.1-I-7B	8	69.60	83.66	77.99	82.46	81.93	81.67	76.44	79.11 (+3.02)
		32	72.13	84.65	79.48	82.83	82.44	81.56	76.83	**79.99** (+3.9)
	ChatGPT	8	70.10	83.92	78.73	82.41	81.96	80.37	77.49	79.28 (+3.19)
		32	72.38	84.62	79.19	82.74	81.85	81.14	77.76	**79.96** (+3.87)
Embedder: Llama2-7B										
Token Avg.†	-	-	35.49	53.15	40.12	55.35	53.26	42.10	49.96	47.06
Echo†			52.40	72.40	61.24	72.67	73.51	65.73	64.39	66.05
PromptEOL†			58.81	77.01	66.34	73.22	73.56	71.66	69.64	70.03
PCoTEOL†			67.45	83.89	74.14	79.47	80.76	78.95	73.33	76.86
KEEOL†			66.60	82.62	74.48	80.75	80.13	80.34	75.89	77.14
MetaEOL†			64.16	81.61	73.09	81.11	79.84	77.96	74.46	75.96
GenEOL	Mistral0.1-I-7B	8	67.37	82.48	76.50	81.33	79.81	80.03	77.68	77.89 (+0.75)
		32	70.24	83.43	78.03	81.79	80.65	80.46	78.08	**78.95** (+1.81)
	ChatGPT	8	68.39	82.48	77.00	82.05	80.28	79.20	78.38	78.25 (+1.11)
		32	70.78	83.28	77.75	82.10	80.45	79.83	78.71	**78.99** (+1.85)
Embedder: Llama3-8B										
PromptEOL†	-	-	60.88	78.57	68.18	76.75	77.16	72.83	68.94	71.90
PCoTEOL			65.38	82.44	71.26	79.22	79.80	77.99	72.49	75.51
KEEOL			62.18	82.35	73.04	80.13	80.17	78.95	77.23	76.29
MetaEOL†			65.10	83.08	73.01	81.57	81.47	80.47	76.46	77.35
GenEOL	Mistral0.1-I-7B	8	68.71	84.24	77.95	82.13	82.44	81.80	78.74	79.43 (+2.08)
		32	71.39	85.13	79.63	82.44	82.79	82.16	79.03	**80.37** (+3.02)
	ChatGPT	8	68.63	84.03	78.44	81.98	82.57	80.45	79.12	79.32 (+1.97)
		32	71.07	84.67	78.90	82.09	82.66	81.15	79.33	**79.98** (+2.63)

Table 2: Performance on 7 STS tasks. GenEOL outperforms every other training free method across different embedders. Two best models are **bold**, next two are underline. (.) is gain on top of previous best (same embedder). PCoTEOL, KEEOL use penultimate layer. All other methods use last layer. †: Numbers reported in prior works.

and Pair Classification) using the appropriate metrics described in Muennighoff et al. (2022).

4.1 Implementation

We explore multiple choices for both generator and embedder. For generator, we try with both small and large models: Mistral0.1-I-7B (Mistral-7B-Instruct-v0.1) and ChatGPT (gpt-3.5-turbo-0125). Embedder is the common module that GenEOL shares with other baselines and becomes the basis of comparison. For embedder, we try three pretrained models based on previous methods - Mistral0.1-7B (Mistral-7B-v0.1), Llama2-7B (Llama-2-7b-hf), Llama3-8B (Meta-Llama-3-8B).

Embedder in GenEOL, by default, uses the embedding prompt from KEEOL (Zhang et al., 2024) i.e. "*The essence of a sentence is often captured by its main subjects and actions, while descriptive terms provide additional but less central details. With this in mind , this sentence : "[X]" means in one word:*"". We evaluate the performance of GenEOL at different values of m. When $m > 4$, multiple transformed sentences are sampled from the generator from each transformation.

4.2 Main Table

Table 2 shows results of multiple methods on STS. Results for KEEOL and PCoTEOL use the penultimate layer as proposed in Zhang et al. (2024). Every other method (including GenEOL) uses the final layer. GenEOL significantly beats the next best method by **(3.88 , 1.83, 2.83)** points on average with (Mistral0.1-7B, Llama2-7B, Llama3-8B) respectively on the 7 STS task average. The average gain of GenEOL (m=32) is a **2.85 points** higher than the previous best method across models. **GenEOL demonstrates consistent performance improvements across all embedder models**, a trend that is uncommon in previous approaches.

For a more fair comparison, we additionally pro-

Method	STS12	STS13	STS14	STS15	STS16	STSB	Sick-R	Avg.
Generator: Mistral0.1-I-7B; $m = 8$; **Embedder**: Mistral0.1-7B;								
GenEOL	69.60	83.66	77.99	82.46	81.93	81.67	76.44	**79.11**
GenEOL w/o Composition	69.02	83.30	77.16	81.61	81.84	80.55	76.52	78.57
GenEOL w/o Composition, w/o $\mathcal{T}_2$, $\mathcal{T}_3$, $\mathcal{T}_4$	68.98	82.54	76.37	81.48	80.73	79.38	76.93	78.06
GenEOL w/o Composition, w/o $\mathcal{T}_1$, $\mathcal{T}_3$, $\mathcal{T}_4$	67.77	82.46	76.48	81.65	81.34	79.21	76.16	77.87
GenEOL w/o Composition, w/o $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_4$	61.69	81.27	74.55	78.23	78.21	72.96	73.22	74.30
GenEOL w/o Composition, w/o $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$	67.68	81.61	75.04	80.76	79.10	78.75	75.84	76.97
GenEOL w/o Composition, w/o $\mathcal{T}_1$, $\mathcal{T}_2$, $\mathcal{T}_3$, $\mathcal{T}_4$	60.25	78.69	69.60	76.79	76.54	75.21	73.50	72.94

Table 3: Dissecting GenEOL. $m = 8$ for all rows. 1. Removing composition hurts performance (Row 2). 2. Non diverse transformations hurt performance (Rows 3-6). GenEOL with diverse transformations, composition is the best.

vide results with GenEOL at $m = 8$ generations. MetaEOL uses eight diverse prompts and averages 8 embeddings of the input sentence. GenEOL averages among $m + 1 = 9$ embeddings. GenEOL ($m = 8$) beats MetaEOL by **2.41 points** on average across models. This shows that "**Diverse meaning retaining transformations can significantly improve sentence embeddings**".

4.3 Ablation 1: Effect of increasing the number of transformed sentences, m

The number of transformed sentences, denoted as m, is a key parameter that significantly influences the performance of GenEOL. To provide a clearer understanding of GenEOL's effectiveness, we present results with varying m values for both the Mistral0.1-7B and Llama3-8B embedding models in Fig. 3. For $m = 2$, we randomly sample from transformations $\mathcal{T}_1$ and $\mathcal{T}_2$. For $m \geq 4$, we maintain and equal diversity across all transformations - ($\mathcal{T}_1$...$\mathcal{T}_4$). **Even with just two generations, GenEOL ($m = 2$) beats all other baselines**.

Increasing m results in a significant increase in performance from $m = 0$ to $m = 32$ of over 5 points. The improvements are lower at higher m. Performance starts to stagnate after $m = 16$. This is because similar transformed sentences/repetitions are sampled from the generator at higher m for a fixed number of transformations (4 in our case). Hence, **more diverse transformations might be required at higher m values**.

4.4 Ablation 2: Dissecting GenEOL

To understand the contribution of different components of GenEOL, we assess performance of individual components of GenEOL. Table 3 shows results. All methods use the exact same value of $m = 8$. Removing compositions results in notable reduction in performance (-0.54). Hence, similar to SumCSE, summary compositions are important.

Among the individual transformations, $\mathcal{T}_1$ (changing sentence structure) performs the best. Using only transformed sentences from $\mathcal{T}_1$ results in a drop of 1.5 points. $\mathcal{T}_3$ (Entailment) performs the worst in this aspect resulting in a drop of 3.7 points. We posit this is because entailment transformation slightly changes the meaning in the transformed sentence. However, this still does better than not using any transformations.

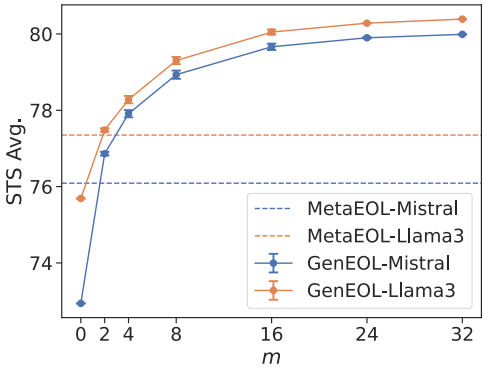


Figure 3: Average STS performance with m. GenEOL beats MetaEOL with only 2 transformations (m=2) for both embedders. Performance starts to stagnate around 16 generations. Results averaged over 3 seeds.

4.5 Ablation 3: Effect of Layer Number

Prior work on training free embeddings has shown that the last hidden layer might not be the most appropriate layer for text embeddings (Lei et al., 2024; Zhang et al., 2024; Li and Li, 2024). We hence evaluate the performance of GenEOL across different hidden layers. We perform the same analysis for KEEOL. Penultimate layer does best for Llama3-8B and second best for Mistral0.1-7B. Performance drops beyond the penultimate layer.

Additionally, the variation (max-min) across layers is lower for GenEOL at (1.52, 1.44) compared to KEEOL at (2.22, 1.89) and MetaEOL at (2.5, unknown) for (Mistral0.1-7B, Llama3-8B) respectively. Hence, **GenEOL can stabilize the representational quality across the LLM layers**.

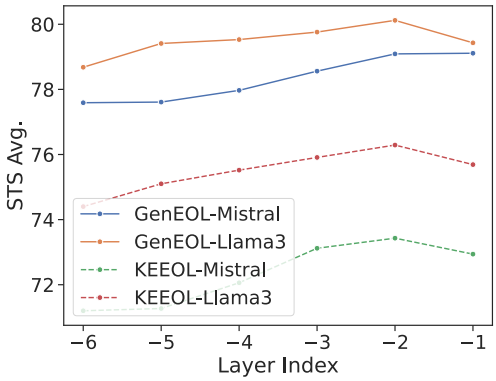


Figure 4: Avg. STS performance when embeddings are extracted from different layers. GenEOL can stabilize the representational quality across layers. Max-Min value across layers is lower for GenEOL compared to KEEOL

4.6 Compute Allocation in GenEOL

Datapoints in STS datasets comprise of a pair of sentences (*sentence*$_1$, *sentence*$_2$) whose similarity needs to be predicted. Given a budget of generating 32 transformed sentences, GenEOL by default invests this budget equally into both sentences in the pair. From the intuition discussed in §3.1, this would create large pool of pairs which result in better averaging and improved results. Alternatively, one might invest this entire generation budget to *sentence*$_1$ or *sentence*$_2$ individually.

In this ablation, we assess the effect of unequally splitting budget between the two sentences. Results are shown in Table 4. Interestingly, allocating all generations to one of the sentences does worse than the baseline (without any generations i.e. $m = 0$). This is counter-intuitive to §3.1, which says that increasing the number of pairs improves performance. We posit this happens because a sentence when averaged with large number of embeddings, undergoes a normalizing effect and starts to focus on the core aspects of the sentence. On the other hand, the sentence which only has one embedding encodes some tail aspects about the sentence. This hurts the cosine similarity score between them. Including a few generations on both sentences performs significantly better i.e. the $(8, 24)$ combinations perform much better than $(0, 32)$ ones. Overall, **equally splitting the generation compute between the two sentences performs best** among the variants.

4.7 Sensitivity to Embedder Prompts

Changing prompts results in high variance for training free methods (Lei et al., 2024). In this subsection, we asses the variance cause by using different prompts to the performance of training free embeddings. We compare our method to a baseline with varying prompts in Table 5. Note all results are with the last hidden layer embeddings.

For the first experiment, we use the three major prompts discussed earlier - PromptEOL, PCoTEOL, KEEOL. Results show that average variance of our method, Gen*EOL, across models is lower than the baseline, *EOL. For the second experiment, we make minimal changes to the KEEOL prompt (removing/adding slack whitespaces) to form the KEEOL′ prompt shown in §A.5. KEEOL′ results in high variance for *EOL, but stable performance (low variance) for Gen*EOL. Thus, **GenEOL is fairly robust to variations in embedding prompts**.

Method	m_1	m_2	STS Avg.
GenEOL	16	16	**79.70**
GenEOL	24	8	79.30
GenEOL	32	0	68.91
GenEOL	8	24	78.96
GenEOL	0	32	69.92
KEEOL (Last Layer)	-	-	72.94

Table 4: Performance of varying the compute on Sentence 1 vs Sentence 2. Spending compute equally gives the highest gains. All compute spent on Sentence 1 or Sentence 2 does even worse than the baseline.

Prompts	Embedder	Gen*EOL	*EOL
PromptEOL, PCoTEOL, KEEOL	Mistral0.1-7B	79.03 ± 0.88	72.31 ± 0.76
	Llama3-8B	79.76 ± 0.88	73.42 ± 3.24
KEEOL, KEEOL′	Mistral0.1-7B	79.98 ± 0.01	71.92 ± 1.44
	Llama3-8B	80.17 ± 0.28	74.48 ± 1.71

Table 5: (Avg. ± Std) across varying prompts. $*$ indicates prompt average. Gen*EOL more stable than *EOL.

4.8 Sensitivity to Generator Prompts

The prompts tried in the §3.2 were few shot prompts emphasizing that the core meaning needs to be retained. In this subsection, we try to understand the sensitivity of such generator prompts. We vary both the generator and the prompt to understand this. In the first experiment in Table 7, we use the exact same prompts and change the generator: we use Llama-I-3.1 (Meta-Llama-3.1-8B). Although significantly better than the baseline from Table 2, this result lags behind using Mistral0.1-I-7B as the generator.

For the second experiment in Table 7, we merge the four few shot prompts and form a single prompt (without any demonstrations) and sample multiple times from it. As expected this drastically reduces performance with Mistral0.1-I-7B due to low

Method	Classification			Clustering		Reranking			Pair Classification	
	AC	B7	EC	MX	TN	AU	SD	SO	TS	SD
KEEOL	72.72	78.70	48.46	24.54	30.48	53.95	73.44	39.12	71.03	43.70
TSEOL	**74.91**	**82.89**	**50.70**	30.88	50.87	62.52	85.26	47.78	73.75	69.18
GenTSEOL($m = 8$)	71.59	82.73	47.54	**31.14**	**51.78**	**63.58**	**85.44**	**48.45**	**76.82**	**77.58**

Table 6: Results on MTEB tasks across 4 categories. TSEOL is a task specific EOL prompts. TSEOL does better than KEEOL for all tasks. GenTSEOL (generator = ChatGPT) does better than TSEOL in all tasks except for the classification.

quality transformations. We perform a small qualitative analysis to understand the results.

Qualitative Analysis: Some examples for each of the four runs from Table 7 are shown in §A.6. Poor quality generations that missed out essential details/hallucinated new content are highlighted. As can be seen, Llama-I-3.1 with few shot prompts and Mistral0.1-I-7B with zero shot prompt produced poor transformations.

Overall, we suggest that tuning prompts on the development set could enhance GenEOL for use with other generator models and prompts. For instance, the compositional summary prompt in Table 8 is derived from tuning its counterpart from SumCSE on the STSB validation set, resulting in an improvement in performance from 78.23 to 79.11 on the STS test set. Interestingly, **ChatGPT does very well with the single zero shot prompt to generate good quality transformed sentences**.

Embedder: Mistral0.1-7B; $m = 8$;		
Prompt Type	**Generator**	**STS Avg**
Four Few Shot Prompts	Mistral0.1-I-7B	79.11
	ChatGPT	79.28
	Llama-I-3.1	77.48
Single Zero Shot Prompt	Mistral0.1-I-7B	76.53
	ChatGPT	79.89

Table 7: Performance of various models under two distinct prompt configurations. ChatGPT demonstrates strong effectiveness even with a single zero-shot prompt.

4.9 More MTEB Datasets

The STS tasks analysed so far are all based on sentence similarity, which is the main focus of this paper. In this subsection, we asses the performance of our method on other tasks from MTEB benchmark (Muennighoff et al., 2022). As our method is slightly expensive, we specifically pick tasks that are tractable for our method. We skip tasks that contain large number of datapoints (E.g. retrieval tasks have millions of documents). Details on specific tasks reviewed in each category are in §A.3.

KEEOL prompt was designed keeping the STS sentences in mind. It might not be suitable for other tasks like clustering medical paper titles. Hence, we first come up with appropriate EOL prompts for the different tasks mentioned. We call this collection - Task Specific EOL (TSEOL) (§A.7). TSEOL outperforms KEEOL across all MTEB tasks tried.

We use the same transformations as described in §3.2 to generate transformed sentences for the above tasks. We use ChatGPT as the generator to asses the performance, minimizing any poor generations. These transformed generations are then used with TSEOL prompts to get numbers for GenTSEOL. GenTSEOL results in notable gains across tasks except for classification.

Classification tasks often rely on specific aspects of a sentence (E.g. 'Emotion' in EC and 'Intent' in B7). These aspects are very specific to the original sentence. Transformations in GenTSEOL often change sentence structure, active speech to passive, etc. and might not retain such specific aspects like emotion. Hence, GenTSEOL performs worse on classification tasks. The proposed GenTSEOL framework allows for other transformations than ones prescribed in §3.2. We believe such task specific transformations can further improve GenTSEOL. We leave this exploration for future work.

5 Conclusion

We introduce GenEOL, an effective method for leveraging the generative capabilities of large language models (including black-box models) to enhance sentence embeddings. Through extensive experiments and ablations, we show that aggregating embeddings of diverse meaning retaining transformations can significantly improve sentence embeddings. Even with very small number of transformed sentences i.e. ($m = 2$), GenEOL beats all baselines. GenEOL stabilizes representational quality across LLM layers and is robust to perturbations of embedding prompt. Despite gains on diverse MTEB tasks, GenEOL falls short on classification tasks requiring identification of specific aspects about original sentence, such as emotion, subjectivity, or intent.

6 Limitations

The main limitation of GenEOL is the cost associated with generating multiple transformations and then embedding them.

GenEOL is generally computationally expensive, since we need to generate between 2 and 32 transformations of a given sentence. However, we note that this is all inference time compute which only requires forward passes. Such inference time compute methods can be more effectively parallelized compared to methods which require a high training compute (backward passes) (Anil et al., 2018; McCandlish et al., 2018; Thirukovalluru et al., 2024a).

Recently, larger models like GPT-4o (Achiam et al., 2023) and LLMonkeys (Brown et al., 2024) have leveraged extra inference time compute to significantly improve reasoning tasks. Our work, GenEOL, is the first such effort to use inference time compute to improve text embeddings.

Moreover, techniques such as speculative decoding (Li et al., 2024) and inference engines such as vllm (Kwon et al., 2023) have made great strides in reducing the cost and latency of sampling from Large Language Models. These methods can be used to make GenEOL more efficient.

Broader Impact and Discussion of Ethics:
While our model is not tied to any specific applications, it could be used in sensitive contexts such as health-care, etc. Any work using our method is requested to undertake extensive quality-assurance and robustness testing before applying in their setting. To the best of our knowledge, the datasets used in our work do not contain any sensitive information.

License: All datasets, methods used fall under Apache License 2.0. This research work abides by terms of the license. Research output of this paper also fall under Apache License 2.0.

Replicability:
Source code will be made public.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Rohan Anil, Gabriel Pereyra, Alexandre Passos, Robert Ormandi, George E Dahl, and Geoffrey E Hinton. 2018. Large scale distributed neural network training through online distillation. *arXiv preprint arXiv:1804.03235*.

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*.

Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

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

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

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626.

Jinhyuk Lee, Zhuyun Dai, Xiaoqi Ren, Blair Chen, Daniel Cer, Jeremy R Cole, Kai Hui, Michael Boratko, Rajvi Kapadia, Wen Ding, et al. 2024. Gecko: Versatile text embeddings distilled from large language models. *arXiv preprint arXiv:2403.20327*.

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*.

Xianming Li and Jing Li. 2024. Bellm: Backward dependency enhanced large language model for sentence embeddings. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 792–804.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*.

Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024. Improving

llm reasoning through scaling inference computation with collaborative verification. *arXiv preprint arXiv:2410.05318*.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. 2018. An empirical model of large-batch training. *arXiv preprint arXiv:1812.06162*.

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*.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. 2022. Mteb: Massive text embedding benchmark. *arXiv preprint arXiv:2210.07316*.

Jacob Mitchell Springer, Suhas Kotha, Daniel Fried, Graham Neubig, and Aditi Raghunathan. 2024. Repetition improves language model embeddings. *arXiv preprint arXiv:2402.15449*.

Raghuveer Thirukovalluru, Nicholas Monath, Bhuwan Dhingra, and Sam Wiseman. 2024a. Sequence reducible holdout loss for language model pretraining. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 14705–14716.

Raghuveer Thirukovalluru, Xiaolan Wang, Jun Chen, Shuyang Li, Jie Lei, Rong Jin, and Bhuwan Dhingra. 2024b. Sumcse: Summary as a transformation for contrastive learning. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 3577–3588.

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

Bowen Zhang, Kehua Chang, and Chunping Li. 2024. Simple techniques for enhancing sentence embeddings in generative language models. In *International Conference on Intelligent Computing*, pages 52–64. Springer.

Junlei Zhang, Zhenzhong Lan, and Junxian He. 2023a. Contrastive learning of sentence embeddings from scratch. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3916–3932, Singapore. Association for Computational Linguistics.

Yan Zhang, Zhaopeng Feng, Zhiyang Teng, Zuozhu Liu, and Haizhou Li. 2023b. How well do text embedding models understand syntax? *arXiv preprint arXiv:2311.07996*.

A Appendix

A.1 Models and Parameters

Model names and parameter counts are as follows. **Generation Models Used**: `Mistral0.1-I-7B` (Mistral-7B-Instruct-v0.1); `ChatGPT` (gpt-3.5-turbo-0125); `Llama-I-3.1` (Meta-Llama-3.1-8B) **Embedder Models Used**: `Mistral0.1-7B` (Mistral-7B-v0.1), `Llama2-7B` (Llama-2-7b-hf), `Llama3-8B` (Meta-Llama-3-8B).

A.2 Compute

All experiments were run on four A6000 GPUs (48gb). Generating transformations for each of the 7 STS datasets using `Mistral0.1-I-7B` took 3 hrs for `GenEOL` ($m = 8$) and 12 hrs for ($m = 32$). `ChatGPT` timings were approximately same.

A.3 Datasets

MTEB: We pick the following tasks-
1. Classification: AmazonCounterFactualClassification (AC), Banking77Classification (B7), EmotionCLassification (EC).
2. Clustering: MedarxivP2P (MX), TwentyNewsClustering (TN).
3. Reranking: AskUbuntuDupQuestions (AU), SciDocsRR (SD), StackOverflowDupQuestions (SO).
4. Pair Classification: TwitterSemEval2015 (TS), SprintDuplicateQuestions (SD).
Dataset sizes: Conneau and Kiela (2018); Muennighoff et al. (2022)

A.4 Transformation Prompts

All transformation prompts used in the work are listed in Table 8. For few shot prompts, demonstrations would follow the given text.

A.5 Sensitivity to Embedder Prompts

§4.7 uses multiple prompts to check the sentitivity of `GenEOL` to embedder prompts. These prompts are listed in Table 11.

A.6 Sensitivity to Generator Prompts

Transformed sentences using the four LM prompt combinations are shown in Table 10. Some poor transformed sentences that either do not retain all aspects from the original sentence or contain some extra content compared to the original sentence are highlighted. `Llama3.1-I-8B` with four few shot prompts and `Mistral0.1-I-7B` have some poor quality transformed sentences.

A.7 Task Specific Prompts for MTEB

§4.9 tuned the EOL prompts to be more applicable to MTEB tasks. Thee tuned prompts collection namely `TSEOL` is shown in Table 12.

Task Code	Description
$\mathcal{T}_1$	Rewrite the input sentence or phrase using different sentence structure and different words while preserving its original meaning. Please do not provide any alternative or reasoning or explanation.
$\mathcal{T}_2$	Create a sentence or phrase that is also true, assuming the provided input sentence or phrase is true. Please do not provide any alternative or reasoning or explanation.
$\mathcal{T}_3$	Provide a concise paraphrase of the input sentence or phrase, maintaining the core meaning while altering the words and sentence structure. Feel free to omit some of the non-essential details like adjectives or adverbs. Please do not provide any alternative or reasoning or explanation.
$\mathcal{T}_4$	Paraphrase the input sentence or phrase, providing an alternative expression with the same meaning. Please do not provide any alternative or reasoning or explanation.
`Composition Summary`	Summarize the input sentence while preserving the exact meaning of the sentence. Do not output any additional explanation. Only output the summary.
`Single Zero Shot Prompt`	Make changes to the following sentence to output 10 diverse sentences. Following are some directions for the changes. 1. Paraphrase the input sentence or phrase, providing an alternative expression with the same meaning. 2. Rewrite the input sentence or phrase using different sentence structure and different words while preserving its original meaning. 3. Create a sentence or phrase that is also true, assuming the provided input sentence or phrase is true. 4. Provide a concise paraphrase of the input sentence or phrase, maintaining the core meaning while altering the words and sentence structure. Feel free to omit some of the non-essential details like adjectives or adverbs. With these possible transformations in mind, transform the following sentence. Output the 10 transformed sentences in 10 new lines. Do not output any explanation. Do not output anything else.

Table 8: For Few shot Prompts for the four transformations. Compositional Summary prompt also mentioned. Few shot demonstrations provided in sourcecode. Single zero short prompt also mentioned.

Transformation	Transformed Sentence
Original Sentence	Aloes, like most succulents, are very hardy and will withstand considerable neglect .
$\mathcal{T}$_1	Like most succulents, aloes are very hardy and can tolerate considerable neglect. Like most succulents, aloes are very hardy and can withstand a considerable amount of neglect.
$\mathcal{T}$_2	Aloes, like many succulents, are very resilient and can tolerate a lot of neglect. Aloes, like most succulents, are hardy and can endure a lot of neglect.
$\mathcal{T}$_3	Succulents are very hardy and will withstand neglect. Aloes are very hardy and can tolerate considerable neglect.
$\mathcal{T}$_4	Aloe plants, similar to other succulents, are extremely resilient and can tolerate a significant level of neglect. Aloes, like many succulents, are extremely resilient and can tolerate a great deal of neglect.

Table 9: Examples of transformed sentences for each transformation with `Mistral0.1-I-7B`. While the transformations themselves are not easily distinguitable, smaller nuances are present within the data. E.g. $\mathcal{T}$_1 sentences have changed sentence structure; $\mathcal{T}$_3 sentences are true statements based on original.

Generator	Transformations
Sentence	Aloes, like most succulents, are very hardy and will withstand considerable neglect .
`ChatGPT` (Single Zero Shot Prompt)	Aloes, similar to many other succulents, can tolerate a lot of neglect. Aloes are extremely resilient plants, much like most succulents. Aloes are known for their ability to thrive even when neglected, a common trait among succulents. Most succulents, including aloes, are quite hardy and can survive neglect. Aloes, along with most succulents, have a high tolerance for neglect. Aloes, as with many succulents, are very resilient and can handle being neglected. Aloes, much like the majority of succulents, are hardy plants that can withstand neglect. Most succulents, such as aloes, are sturdy and can withstand neglect.
`Mistral0.1-I-7B` (Four Few Shot Prompts)	Aloe plants, similar to other succulents, are extremely resilient and can tolerate a significant level of neglect. Aloes, like many succulents, are extremely resilient and can tolerate a great deal of neglect. Like most succulents, aloes are very hardy and can tolerate considerable neglect. Like most succulents, aloes are very hardy and can withstand a considerable amount of neglect. Succulents are very hardy and will withstand neglect. Aloes are very hardy and can tolerate considerable neglect. Aloes, like many succulents, are very resilient and can tolerate a lot of neglect. Aloes, like most succulents, are hardy and can endure a lot of neglect.
`Llama3.1-I-8B` (Four Few Shot Prompts)	Like most succulents, aloes are extremely resilient and can tolerate significant neglect. Like most succulents, aloe plants are extremely resilient and can tolerate significant neglect. Succulents, including aloes, are known for their remarkable hardiness and ability to thrive even when neglected. Succulents, including aloes, are known for their remarkable hardiness and ability to thrive despite considerable neglect. Many plants can tolerate some level of neglect. Most succulents are very hardy plants. Succulents, including aloes, are resilient plants. Succulents like aloes are very resilient.
`Mistral0.1-I-7B` (Single Zero Shot Prompt)	Aloes are hardy like most succulents and can tolerate significant neglect. Most succulents, including aloes, are hardy and can withstand considerable neglect. Aloe vera, like most succulents, is very hardy and can endure considerable neglect. Like most succulents, aloes are extremely hardy and can withstand significant neglect. Aloe vera, being a succulent like most others, is very hardy and can tolerate considerable neglect. Aloe vera, like the majority of succulents, is extremely hardy and can endure considerable neglect. Aloe vera is very hardy, like most succulents, and can withstand considerable neglect. Like most succulents, aloe vera is hardy and can endure significant neglect with ease.

Table 10: Some bad transformations are highlighted. They have one of the two problems - 1. Not all information about the original sentence is retained. 2. LLM hallucinates some content.

Prompt Version	Prompt Text
PromptEOL	This sentence : "{input_text}" means in one word:"
PCoTEOL	After thinking step by step , this sentence : "{input_text}" means in one word:"
KEEOL	The essence of a sentence is often captured by its main subjects and actions, while descriptive terms provide additional but less central details. With this in mind, this sentence: "{input_text}" means in one word:"
KEEOL′	The essence of a sentence is often captured by its main subjects and actions, while descriptive terms provide additional but less central details. With this in mind , this sentence : "{input_text}" means in one word:"

Table 11: Table containing all prompts used in the work. KEEOLand KEEOL′ differ only by two whitespaces.

Task Name	Description
AmazonCounterfactual Classification	In this task, classify a given Amazon customer review text as either counterfactual or not-counterfactual. For this task, this review: "{input_text}" means in one word:
Banking77Classification	In this task, given an online banking query, find the corresponding intents. For this task, this query: "{input_text}" means in one word:
EmotionClassification	In this task, classify the emotion expressed in the given Twitter message into one of the six emotions: anger, fear, joy, love, sadness, and surprise. For this task, this message: "{input_text}" means in one word:
MedrxivClusteringS2S	In this task, identify the main category of Medrxiv papers based on the titles. For this task, this title: "{input_text}" means in one word:
TwentyNewsgroups Clustering	In this task, identify the topic and theme of the news article. For this task, this article: "{input_text}" means in one word:
AskUbuntuDupQuestions	In this task, you need to retrieve duplicate questions from AskUbuntu forum. For this task, this question: "{input_text}" means in one word:
SciDocsRR	In this task, given a title of a scientific paper, retrieve the titles of other relevant papers. For this task, this title: "{input_text}" means in one word:
StackOverflowDup Questions	In this task, retrieve duplicate questions from StackOverflow forum. For this task, this question: "{input_text}" means in one word:
TwitterSemEval2015	In this task, retrieve tweets that are semantically similar to the given tweet. For this task, this tweet: "{input_text}" means in one word:
SprintDuplicateQuestions	In this task, retrieve duplicate questions from Sprint forum. For this task, this question: "{input_text}" means in one word:

Table 12: TSEOL prompts