SEMANTIC FRAGMENT SIMILARITY REPRESENTATION LEARNING FOR INFORMATION RETRIEVAL

Anonymous authors
Paper under double-blind review

ABSTRACT

We introduce Semantic Fragment Similarity (SFS), a similarity metric designed to enhance representation quality by partitioning embeddings into non-overlapping fragments, computing fragment level similarity, and aggregating these local scores. Conventional similarity metrics compute relevance using the global vector as a single unit. This process flattens and entangles multi-faceted semantic features and dilutes the fine-grained alignment signals crucial for accuracy. By inducing fragments to specialize in distinct semantic roles, SFS drives the substantial gains in retrieval performance across a wide range of models, tasks, and architectures when applied in both training and inference. Further, we find that a single embedding fragment trained with SFS, comprising just 2% of the total dimensions, outperforms the entire global embedding on specific classification tasks. Ultimately, SFS can be directly integrated as a replacement for conventional similarity metrics, without architectural modifications or complex computational overhead and it opens up new avenues for building more structured and interpretable embedding models.

1 INTRODUCTION

As information retrieval advances, dense retrieval has become a widely used approach for efficiently retrieving the documents most relevant to a user query from large-scale collections (Karpukhin et al., 2020; Xiong et al., 2020). These approaches encode queries and documents as high-dimensional vectors and retrieve documents by their similarity in the embedding space. This representation captures complex semantic relationships beyond keyword matching (Khattab & Zaharia, 2020), leading to significant performance in applications such as retrieval-augmented generation (RAG) (Lewis et al., 2020; Gao et al., 2023; Jiang et al., 2023).

Dense retrieval models generate an embedding vector that encodes the overall meaning of a text into a single vector representation (Reimers & Gurevych, 2019; Li et al., 2020). The relevance of a query-document pair is quantified by a scalar score, typically derived from a similarity metric such as cosine similarity or the dot product computed between their vector representations (Günther et al., 2024; Lee et al., 2025b). This macro-level computation on the global vector effectively captures the overall meaning, but the process of reducing multifaceted semantics into a single scalar score inherently causes the flattening of local and multi-faceted object signals (Sinha et al., 2024). In this single-vector representation, different semantic features become entangled across the embedding's dimensions, consequently causing these distinct signals to interfere with or neutralize one another (You, 2025). For instance, if a query and a document exhibit a pronounced correspondence on one semantic axis but are irrelevant on another, the global similarity score can dilute this partial alignment, failing to capture the corresponding micro-level alignment in the vector space.

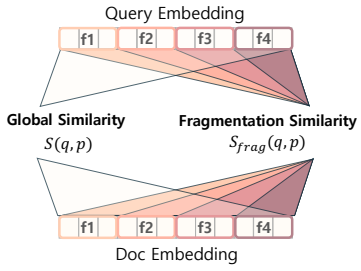


Figure 1: Conceptual Illustration of the Semantic Fragment Similarity.

To mitigate representation flattening and entanglement, we propose Semantic Fragment Similarity (SFS), a novel metric that enhances the measurement of textual relevance by preserving local and multi-faceted semantic signals. To encourage embeddings to encode distinct semantic aspects in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

different subspaces, SFS divides a high-dimensional vector into several non-overlapping fragments. It then calculates similarity for each query-document fragment pair and aggregates these local scores into a final similarity score. This approach allows SFS to explicitly retain fragment-level information, enabling a more fine-grained assessment of textual relatedness while still capturing global semantic context.

To demonstrate the effectiveness of our methodology, we conduct a comprehensive set of experiments applying SFS with diverse fragment size configurations to both embedding model training and retrieval phase. In these experiments, we train five transformer encoder models on retrieval data and conduct extensive evaluations on retrieval tasks in the MTEB benchmark (Muennighoff et al., 2023). The results show that the proposed SFS consistently outperforms a conventional learning approach based on global-vector calculation, yielding substantial improvements in retrieval performance with an average increase of 1–3% over the baseline. Moreover, these gains are observed across diverse embedding tasks (e.g., reranking and classification), pooling strategies, and model architectures, as partitioning into fine-grained fragments preserves and exploits local and multidimensional semantic features that would otherwise be diluted and flattened in a global-vector representation. Our analysis of fragment-level classification performance reveals that SFS encourages fragments to assume distinct semantic roles. This leads to cases where a single fragment, constituting just 2% of the total dimensions, outperforms the full embedding on certain classification tasks. Additionally, we show that SFS can be reduced to standard dot-product operations via a simple transformation, enabling seamless integration with ANN libraries for large-scale retrieval.

We summarize our contributions as follows:

- We propose Semantic Fragment Similarity (SFS), a novel metric that mitigates the flattening and entanglement of semantic signals by aggregating fragment-wise similarities. This method requires no changes to the existing model architecture, such as adding new layers, parameters, or complex computations.
- We experimentally demonstrate that SFS achieves substantial performance gains in our primary task, retrieval, while also consistently outperforming conventional global similarity across other tasks such as classification and reranking.
- Our analysis reveals that SFS induces feature disentanglement and a semantic division of labor among fragments. We identify this mechanism as the driving force behind the observed improvements, offering a new lens that opens up possibilities for the interpretability of the embedding space.

2 Related Work

One factor that undermines the expressiveness of embedding models and degrades retrieval performance is representation dilution. A major cause of representation dilution arises in the pooling process that aggregates token-level representations into a single document vector. Pooling inevitably incurs information loss as it compresses the rich semantic content of the input text into a single vector. In this regard, Lee et al. (2025a) point out that mean pooling dilutes information about salient spans, and that [EOS] pooling can suffer from recency bias (Lin et al., 2025), thereby clarifying the limitations of both approaches. This dilution problem becomes more severe as text length increases: Hu et al. (2025) note that representing a long context with a single vector can lead to representation collapse (Zhou et al., 2024) or dilution. Furthermore, structural characteristics of the model, such as causal attention, can also induce dilution of discriminative information in the latter part of the text (Springer et al., 2024).

Beyond these characteristics, the similarity metric itself also acts as a key contributing factor. In practice, many recent embedding models adopt cosine similarity as the primary similarity measure (Günther et al., 2024; Lee et al., 2025b; Muennighoff et al., 2024; Choi et al., 2024). In particular, cosine similarity is directly included in the training objective of contrastive learning, such as the InfoNCE loss, playing a decisive role in shaping the model's representation space. However, cosine similarity has several known limitations. Its reliability over learned embeddings has been questioned, and cosine-based objectives may be less effective at modeling subtle semantic differences (Steck et al., 2024; Li & Li, 2024). Cosine similarity can also give rise to anisotropy in the embedding space (You, 2025; Liang et al., 2021). This issue causes most vectors to cluster within a narrow cone

region in high-dimensional space, undermining semantic distinction, and inducing a distance concentration effect where pairwise distances become similar, thus degrading the reliability of retrieval systems (Wang & Isola, 2020). Additionally, other metrics such as Euclidean distance can flatten complex hierarchical semantic relations, distorting the original semantic context (Sinha et al., 2024).

3 SEMANTIC FRAGMENT SIMILARITY

In this section, we first introduce the training methodology for dense retrieval and the notation used throughout this paper. We then describe the training and inference processes using our proposed Semantic Fragment Similarity in detail.

3.1 PRELIMINARIES

In a dense retrieval framework using an embedding model, the objective is to retrieve semantically relevant documents from a large corpus $\mathcal{D} = \{d_1, d_2, \ldots, d_N\}$ given a query q. An embedding model $\mathcal{M}$ encodes textual inputs into a high-dimensional vector space of a fixed dimension. The model's output is then processed through a pooling operation to produce a dense vector in $R^{d_{model}}$, where d_{model} denotes the model's output embedding dimension. For a given query q and a document $d \in \mathcal{D}$, their embeddings are generated as follows:

$$\mathbf{E}_q = \text{Pooling}(\mathcal{M}(q)), \quad \mathbf{E}_d = \text{Pooling}(\mathcal{M}(d)) \in \mathbb{R}^{d_{\text{model}}} \tag{1}$$

The semantic relevance between a query q and a document d is quantified by a score $S(q, d)$, which is typically defined as the cosine similarity between their embedding vectors E_q and E_d. This similarity function aggregates information across all dimensions of the representations and returns a single scalar that reflects the overall semantic similarity between the query and the document.

$$S(q, d) = \frac{\mathbf{E}_q \cdot \mathbf{E}_d}{\|\mathbf{E}_q\| \, \|\mathbf{E}_d\|} \tag{2}$$

The embedding model is trained to optimize retrieval performance using a contrastive learning objective. The goal is to maximize similarity score $S(q, d^+)$ for relevant query–positive document pairs (q, d^+) and minimize the similarity score $S(q, d^-)$ for non-relevant query–negative document pairs (q, d^-).

3.2 FRAGMENT REPRESENTATION LEARNING

Our method, Semantic Fragment Similarity, learns to produce a final score by aggregating similarity scores calculated for each pair of corresponding fragments. These fragments are obtained by independently assessing the subspace correlation between two vectors within more granular subspaces, each representing a specific semantic aspect.

Fine-grained Semantic Fragmentation The objective of SFS is to produce a single scalar value that reflects micro-level alignment within the vector space. It does so by distinguishing strong signals confined to certain dimensions from weak signals dispersed across multiple dimensions.

We first define a fragmentation function $\mathcal{F}$ that partitions an embedding vector $E \in \mathbb{R}^{d_{\text{model}}}$ into an ordered sequence of non-overlapping fragments. Given a fragment dimension d_{frag}, the total number of fragments N_f is determined as $d_{\text{model}}/d_{\text{frag}}$. The fragmentation function then maps the vector E to a fragment sequence $\mathcal{F}(E) = \{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{N_f}\}$, where each fragment $\mathbf{f}_i$ has a fixed dimension $\mathbb{R}^{d_{\text{frag}}}$. The fragment size d_{frag}, specified externally as a hyperparameter, ensures that all fragments have identical dimensionality. For instance, if the embedding dimension $d_{\text{model}} = 768$ and the fragment dimension $d_{\text{frag}} = 64$, the fragmentation function produces a sequence of 12 fragments $\{\mathbf{f}_1, \mathbf{f}_2, \ldots, \mathbf{f}_{12}\}$, each of dimension 64. Concretely, the first fragment $\mathbf{f}_1$ comprises the components of E from indices 0 to 63. More generally, the i-th fragment $\mathbf{f}_i$ corresponds to the slice $E[(i-1) \times d_{frag} : i \times d_{frag}]$.

Fragment-wise Similarity Aggregation The query embedding E_q and the document embedding E_d are each partitioned into a sequence of N_f fragments, $(\mathbf{f}_{q1}, \mathbf{f}_{q2}, \ldots, \mathbf{f}_{qN_f})$ and $(\mathbf{f}_{d1}, \mathbf{f}_{d2}, \ldots, \mathbf{f}_{dN_f})$,

respectively. Instead of a global operation, we independently compute the cosine similarity, $s_i(q, d)$, for each corresponding fragment pair $(\mathbf{f}_{qi}, \mathbf{f}_{di})$ and then average the fragment-wise similarity scores to obtain a single similarity score $S_{\text{frag}}(q, d)$.

$$S_{\text{frag}}(q, d) = \frac{1}{N_f} \sum_{i=1}^{N_f} s_i(q, d) = \frac{1}{N_f} \sum_{i=1}^{N_f} \cos\big(\mathbf{f}_{qi}, \mathbf{f}_{di}\big) \tag{3}$$

This aggregation scheme equally weights alignment across semantic aspects, thereby preventing the flattening of information that arises during encoding and resulting in a final score that reflects micro-level alignment across the entire embedding space.

3.3 Training and Inference

Training To apply our proposed SFS in training, we integrate it into a contrastive objective. Specifically, within the infoNCE loss, we replace the conventional similarity function $S(q, d)$ with our $S_{\text{frag}}(q, d)$. The training objective is to minimize the following loss function:

$$L_{\text{NCE}} = -\frac{1}{n} \sum_{i} \log \frac{\exp(S_{\text{frag}}(q_i, d_i^+))}{\exp(S_{\text{frag}}(q_i, d_i^+)) + \sum_j \exp(S_{\text{frag}}(q_i, d_j^-))} \tag{4}$$

By minimizing this loss, the model learns to maximize fragment level semantic alignment for positive pairs (q, d^+) and minimize it for negative pairs (q, d^-). This training objective induces the model to structure the entire embedding space into a set of semantically coherent subspaces. Instead of relying on a single, global representation, the model learns to encode distinct semantic facets (e.g., topic, domain, and style) into different fragments of the embedding vector. Consequently, each fragment becomes more adept at comparing these subtle and local semantic features, which leads to a more robust and interpretable similarity measure compared to a conventional single scalar value.

Inference At inference time, we encode the query and all documents into their respective embeddings. Subsequently, the final document ranking is determined using the S_{frag}. Algorithm 1 summarizes the specific inference procedure for ranking the entire document corpus for a given query. Furthermore, we discuss how the proposed SFS can be integrated with standard Approximate Nearest Neighbor (ANN) search libraries for efficient large-scale retrieval. Appendix B provides detailed explanations on this integration and the corresponding implementation details.

Algorithm 1 Inference with SFS

Input: Query embedding $\mathbf{E}_q$, Document embeddings $\{\mathbf{E}_{d_j}\}_{j=1}^{N}$, Fragment size d_{frag}
Output: Ranked documents
$N_f \leftarrow d_{\text{model}}/d_{\text{frag}}$
$\{\mathbf{f}_{qi}\}_{i=1}^{N_f} \leftarrow \mathcal{F}(\mathbf{E}_q)$
for $j \leftarrow 1$ **to** N **do**
  $\{\mathbf{f}_{d_j i}\}_{i=1}^{N_f} \leftarrow \mathcal{F}(\mathbf{E}_{d_j})$
  $S_{\text{frag}}(\mathbf{E}_q, \mathbf{E}_{d_j}) \leftarrow \frac{1}{N_f} \sum_{i=1}^{N_f} \cos(\mathbf{f}_{qi}, \mathbf{f}_{d_j i})$
end
return sort(Documents, by=S_{frag})

4 Experiments

4.1 Setup

Training Our primary experiments focus on transformer encoder architectures. We select five models, all with a 768-dimension hidden state: ModernBERT-base (Warner et al., 2025), bert-base-uncased (Devlin et al., 2019), gte-en-mlm-base (Zhang et al., 2024), roberta-base (Liu et al., 2019), and nomic-bert-2048 (Nussbaum et al., 2025). We employ the [CLS] pooling strategy to generate sentence embeddings. For each fragment size in $d_{\text{frag}} \in \{768, 64, 32, 16\}$, we train a dedicated model and subsequently evaluate it under the same size configuration. Specifically, the $d_{\text{frag}} = 768$ setting corresponds to the conventional training paradigm where the entire hidden state is treated as a single vector, serving as our baseline. In all experiments, configurations are denoted as (number of fragments, fragment size).

We utilize the standard InfoNCE (van den Oord et al., 2018) loss over in-batch negatives and hard negatives (Henderson et al., 2017). For training data, we employ a collection of publicly available datasets that are standard training datasets in the embedding literature. Further details regarding our overall implementation and the training datasets are provided in Appendix C.

Table 1: Retrieval performance (nDCG@10) by fragment size. Configurations are denoted as (Number of Fragments, Fragment Size). **Bold** and underlined values are the best and second-best scores for each model, respectively.

Model	Global	Semantic Fragment Similarity		
	(1,768)	(12,64)	(24,32)	(48,16)
BERT	0.5178	0.5163	**0.5197**	0.5181
RoBERTa	0.5045	0.5186	0.5168	**0.5195**
NomicBERT	0.5356	0.5414	0.5414	**0.5427**
ModernBERT	0.5375	0.5257	0.5483	**0.5512**
GTE-en-MLM	0.5424	0.5500	**0.5517**	0.5508

Evaluation We conduct evaluations on the Massive Text Embedding Benchmark (MTEB) (Muennighoff et al., 2023). Our experiments focus on 12 retrieval datasets, using nDCG@10 as the primary evaluation metric. All evaluations are conducted using the official MTEB to ensure consistency and reproducibility. A comprehensive list of the datasets included in our evaluation is available in Appendix D.

4.2 Results

Table 1 presents retrieval performance across five models with varying fragment sizes. It shows that our proposed Semantic Fragment Similarity consistently outperforms similarity computations based on global vectors. Across all models evaluated in the experiments, partitioning the vectors into multiple fragments to compute similarity yields better results than the conventional single-fragment setting of (1,768).

Notably, all five models consistently outperform the $d_{\text{frag}} = 768$ baseline when utilizing fragmented embeddings, with peak performance typically observed at fragment sizes of 16 or 32. For example, the ModernBERT demonstrated the most significant improvement, achieving approximately a 2.5% gain over the baseline when the fragment size was set to 16. These findings indicate that our approach partitions the high dimensional vector space into multiple fine grained semantic subspaces, evaluates local similarities within each subspace independently, and subsequently aggregates the resulting similarity measures, enabling more effective capture of the multifaceted and subtle relationships.

5 How Robust Is Fragment Representation Learning in Retrieval?

In this section, we conduct in-depth experiments to assess whether representation learning with the proposed Fragment Similarity remains effective under varying conditions in retrieval tasks. Specifically, we systematically and comprehensively explore variations in pooling strategy, model architecture, and similarity metrics.

5.1 Effectiveness Across Different Pooling Methods

Setup To assess robustness across pooling strategies, we replace [CLS] pooling with average pooling. Average pooling generates embeddings by averaging all token representations from the last hidden outputs. We train two models, ModernBERT and BERT, under the same experimental settings as the [CLS] configuration, except for the pooling strategy.

Results As shown in Table 2, overall performance with average pooling is slightly lower or on par with that obtained [CLS] pooling. Crucially, we observe consistent gains over the full-embedding baseline (1,768). For example, ModernBERT-base improves from 0.5390 to as high as 0.5444, consistent with the improvement trend observed under [CLS] pooling. This suggests that average pooling can, via fragment-wise consensus across subspaces, mitigate the influence of outlier token representations and, in turn, yield more robust representations.

Table 2: Performance comparison of different fragment sizes with average pooling.

Model	Global	SFS		
	(1,768)	(12,64)	(24,32)	(48,16)
BERT	0.5215	**0.5241**	0.5211	0.5232
ModernBERT	0.5390	0.5323	**0.5444**	0.5417

5.2 Robustness Across Model Architectures

Setup We evaluate our method on decoder architectures by training two models, Qwen3-0.6B (Team, 2025) and Llama-3.2-1B, which have a hidden dimension of 2048. We utilize LLMs with causal attention, appending an [EOS] token at the end of the input sequence (Zhang et al., 2025). The sentence embedding is derived from the hidden state of the last layer corresponding to this [EOS] token.

Table 3: Retrieval performance of different fragmentation configurations on decoder models.

Model	Global	SFS			
	(1,2048)	(16,128)	(32,64)	(64,32)	(128,16)
Qwen3-0.6B	0.5762	0.5842	**0.5888**	0.5847	<u>0.5874</u>
Llama-3.2-1B	0.5939	0.5953	<u>0.5983</u>	0.5980	**0.6028**

Results Table 3 shows a common trend that learning via fragment similarity yields consistent performance improvements over the baseline for decoder architectures. Both models significantly outperform the baseline with more granular fragmentation, with Qwen3-0.6B achieving a top score of 0.5888 ($d_{\text{frag}} = 64$) and Llama-3.2-1B a top score of 0.6028 ($d_{\text{frag}} = 16$). These results suggest that, similar to encoder models, the embedding space of decoder models is also composed of mixed, functionally distinct features. Our approach effectively separates these antagonistic signals to enhance local alignment, resulting in superior representational quality.

5.3 Effectiveness Across Similarity Metrics

Setup We conducted an experiment to determine if the benefits of fragment-based learning generalize beyond cosine similarity. Using the ModernBERT-base model, we compared a full-embedding baseline (1,768) against a fragmented configuration (48,16) across four metrics: Cosine, Euclidean (L2), Manhattan (L1), and Dot product.

Table 4: Performance comparison of different similarity metrics, with and without Fragment Similarity.

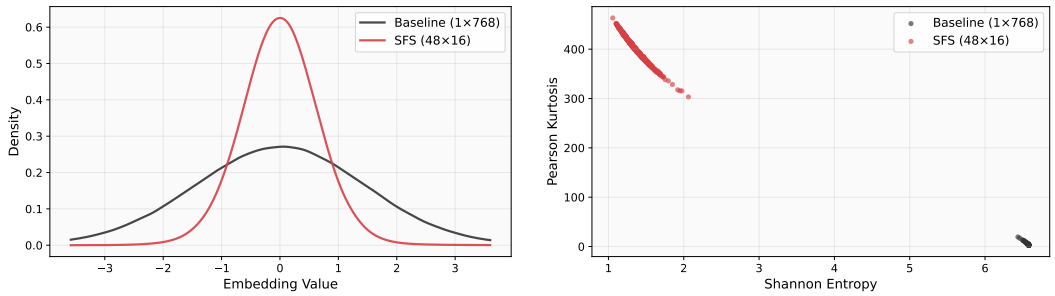
Metric	Global	SFS
	(1,768)	(48,16)
Cosine	0.5375	**0.5512**
Euclidean	0.3348	**0.4680**
Manhattan	0.3582	**0.5145**
Dot	0.3463	**0.5143**

Results Table 4 demonstrates that our Fragment Similarity-based learning is effective for similarity metrics in addition to cosine. Applying fragmentation improves performance across all metrics. Notably, the improvements are substantially larger for metrics sensitive to vector magnitude, such as Euclidean (L2), Manhattan (L1), and the dot product. These results suggest that our method effectively mitigates the problem of anisotropy in the embedding space. The process of measuring and aggregating similarity at the fragment level reduces the directional bias and scale sensitivity of the full vector representation, which we identify as the key reason for the more significant performance enhancements in these metrics.

6 Fragment-Level Analysis and Comparative Evaluation

6.1 Do Fragments Mitigate Representation Flattening?

Setup To investigate the representation flattening effect, we analyze the encoded vectors from both the baseline and the SFS-trained representations using the MIRACL (Zhang et al., 2023a) train dataset. Representation flattening refers to a phenomenon in which representational signals become broadly dispersed across dimensions. To diagnose this effect, we evaluate the embeddings using two complementary analytical approaches. First, to visually assess the continuous distribution and sharpness of the individual dimension values without binning bias, we apply Kernel Density Estimation (KDE) (Silverman, 2018). Second, to simultaneously quantify information concentration and the overall sparsity of the embedding space, we compute both the Shannon entropy and the Pearson kurtosis for each encoded vector. Specifically, we treat each vector as a probability distribution via softmax to calculate its entropy, quantifying the degree of information dispersion. We then pair this with kurtosis to evaluate the presence of extreme outliers, thereby characterizing the overall feature sparsity. Further implementation details for the flattening analysis are provided in Appendix E.



(a) Embedding Coordinate Density. (b) Vector-level Entropy and Kurtosis.

Figure 2: **Analysis of representation flattening.** The figure compares encoded vectors from the global-similarity baseline (1,768) and the SFS-trained representation (48,16). (a) presents the Kernel Density Estimation (KDE) of embedding coordinate values across all samples and dimensions. (b) shows the scatter plot of Shannon entropy versus Pearson kurtosis for individual encoded vectors.

Results Figure 2 illustrates a clear structural difference between the two representation spaces. As shown in Figure 2a, the baseline exhibits a broad and gentle KDE curve, suggesting that its coordinate values are more widely dispersed across the embedding space. Conversely, the representation trained via SFS forms a sharp, high-density peak centered around zero, indicating that most coordinates remain close to zero while a smaller subset of coordinates takes relatively larger values. This structural shift is further supported by the quantitative analysis in Figure 2b, where the SFS-trained representations show a dramatic reduction in entropy (dropping to a range of 1 to 2) coupled with a drastic increase in kurtosis (> 300). This steep decline in entropy and high kurtosis suggest that, rather than dispersing information uniformly across all dimensions, training via SFS concentrates information into a sparse subset of highly active dimensions. By concentrating these discriminative signals into localized fragments, SFS forms a more structured and discriminative representation, providing evidence that fragment-level training mitigates representation flattening.

6.2 Do Fragments Learn Distinct Semantic Roles?

In this section, we experimentally analyze whether training with the proposed methodology indeed induces each fragment of the embedding to assume semantically differentiated roles (distinct semantic roles). Using a ModernBERT trained under various fragment configurations, we conduct two complementary experiments. In the first experiment, we directly compare the classification performance of the full embedding and individual fragments. In the second experiment, we analyze, from an in-depth perspective, how specific fragments respond to specific semantic attributes (semantic features).

6.2.1 Validating Specialization via Classification Performance

Setup We evaluate on two classification datasets with clearly delineated topics, Toxic Conversation and Amazon Counterfactual. We first measure performance using the full embedding, and then assess the performance contribution of individual fragments by comparing it to the performance obtained when using each fragment vector alone.

Results Table 5 shows that as fragment partitioning becomes more fine-grained, classification performance improves consistently. Relative to the baseline, Toxic Conversation increases by up to 11.48%, and Amazon Counterfactual by up to 7.51%. To probe whether the improvements are driven by overall embedding quality or by the semantic specialization of individual fragments, Figure 3 compares the performance of each fragment with that of the full vector. As the number of partitions increases, some fragments clearly outperform the full vector; in particular, in the Toxic

Table 5: Classification performance across different fragmentation configurations on the Toxic Conversation and Amazon Counterfactual datasets.

Dataset	Global	SFS		
	(1, 768)	(12, 64)	(24, 32)	(48, 16)
Toxic Conversation	0.4737	<u>0.5165</u>	0.5126	**0.5281**
Amazon Counterfactual	0.5296	0.5468	**0.5694**	<u>0.5686</u>

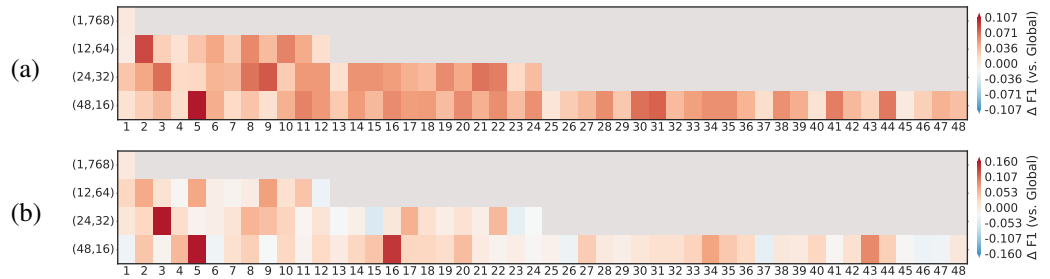
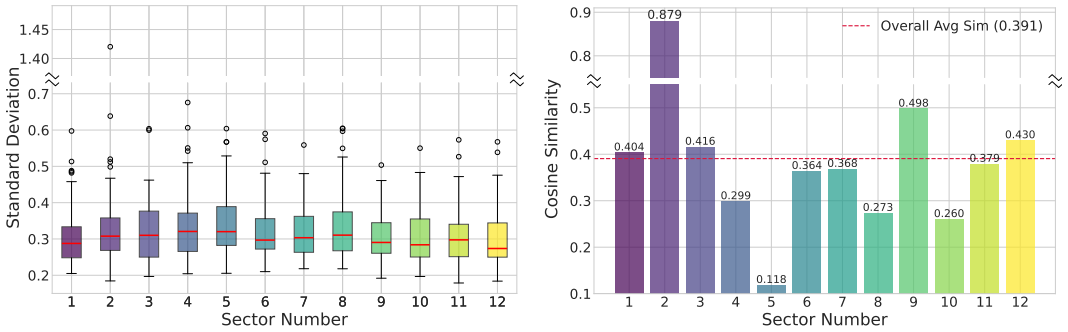


Figure 3: Visualization of single-fragment classification performance relative to the global embedding. Results are shown for (a) the Toxic Conversation and (b) the Amazon Counterfactual.

Conversation (48,16) model, several fragments surpass the whole, and the 5th fragment exceeds it by more than 0.1. In other words, using only 16 dimensions (about 2% of 768) can outperform the full embedding, suggesting that training with fragment similarity concentrates semantic signals within fragments.



(a) Standard Deviation Distribution per Fragment.

(b) Average Cosine Similarity per Fragment.

Figure 4: **Analysis of fragment-level responses to the semantic attribute of toxicity.** (a) displays the distribution of dimension-wise standard deviations for the embedding difference vectors between toxic/neutral sentence pairs, grouped by fragment. (b) shows the average cosine similarity for each fragment between the sentence pairs.

6.2.2 Analyzing the Specialization Mechanism

Setup Building on the previous analysis, we statistically demonstrate how specific fragments encode semantic attributes. In this experiment, we use a model trained with 12 fragments of 64 dimensions each and the TextDetox (Dementieva et al., 2025) dataset containing 400 Toxic/Neutral sentence pairs. To quantify fragment-wise activation differences, we compute the dimension-wise standard deviation of the embedding difference vectors between sentence pairs, group these statistics by fragment and analyze their distributions; in parallel, we measure the average cosine similarity between the fragment embeddings of the sentence pairs.

Results The most notable finding is observed in the second fragment. While this fragment exhibits the highest average cosine similarity across toxic/neutral sentence pairs (Figure 4(b)), it concurrently achieves superior toxicity classification performance for the (12,64) model, as depicted in Figure 3(a). This result is attributable to the characteristics shown in Figure 4(a): a small subset of dimensions within this fragment presents substantial standard deviations in the pairwise difference vectors. This indicates that a minority of dimensions robustly encodes the distinction between toxic and neutral representations, while the majority remain largely invariant. Collectively, these observations validate that our methodology induces fragments to specialize in encoding distinct semantic attributes.

6.3 Comparison with Matryoshka Representation Learning

Setup We compare the efficiency of representation with Matryoshka Representation Learning (MRL) (Kusupati et al., 2022), a representative compression approach for adaptive-dimensional embeddings. MRL concentrates information importance in the prefix of the embedding so that strong performance can be achieved with only a subset of dimensions. For MRL, we train ModernBERT at embedding sizes [768, 512, 384, 256, 128, 64]. For our method, we use a model trained with 48 fragments of 16 dimensions each. For a fair comparison at the same dimensionality, we construct our representation by concatenating fragments of size 16 in order until the target dimension is reached. For example, the 256 dimensional variant uses the first 16 fragments and is compared directly to MRL at 256 dimensions.

Results Figure 5 presents a comparison of retrieval performance across representation sizes. Both methods show a common pattern of improvement as dimensionality increases. For dimensionalities at or below 128, MRL holds a slight advantage, which we attribute to its training objective that concentrates critical information in the prefix of the embedding. However, after the curves cross at 256 dimensions, our method consistently outperforms MRL at all larger sizes. In particular, at 512 dimensions we observe approximately a 4.34% gain in nDCG@10, and the gap continues to widen as dimensionality increases. These results suggest that, unlike MRL, which compresses information into a subset of dimensions, our approach regards each fragment as an independent semantic unit and leverages complementary interactions across fragments to improve representational quality. In addition, these findings suggest that fragment training based on fragment similarity can also serve an adaptive dimensionality.

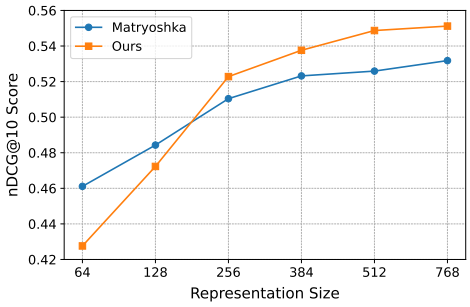


Figure 5: Retrieval Performance Comparison with Matryoshka Representation Learning

6.4 Evaluation on Non-Retrieval Embedding Tasks

Table 6 presents the performance variations in reranking and classification tasks across different fragment sizes, using the full (1,768) embedding as the baseline. Performance on both tasks steadily improves as fragments become more granular; in classification, ModernBERT achieves a performance gain of approximately 2.4%, while BERT also shows consistent enhancements up to the (48,16) configuration. This suggests that semantic fragmentation enables embeddings to effectively represent local semantic information and provides a robust approach to improving performance across diverse downstream tasks.

Table 6: Performance comparison on Reranking and Classification tasks.

Model	**Global**	**SFS**		
	(1,768)	(12,64)	(24,32)	(48,16)
Reranking				
BERT	0.4972	0.4984	0.5004	**0.5007**
ModernBERT	0.5260	0.5260	0.5283	**0.5303**
Classification				
BERT	0.5444	0.5476	0.5479	**0.5487**
ModernBERT	0.5483	0.5507	0.5586	**0.5613**

7 Conclusion

In this paper, we propose Semantic Fragment Similarity (SFS) to address how global similarity calculations in Dense Retrieval flatten multifaceted semantic information, which degrades representation quality. SFS operates by partitioning an embedding vector into multiple non-overlapping fragments, independently computing similarity at the fragment level, and aggregating these local scores. This approach induces the model to assign distinct semantic roles to each fragment, and our experiments demonstrate that it consistently outperforms conventional global similarity calculation across a diverse range of models and tasks. Notably, this approach induces a semantic division of labor, evidenced by our analysis where a single fragment can outperform the entire embedding on specific tasks. Ultimately, SFS challenges the conventional paradigm of relying on global-level operations, presenting a new direction for more structured and reliable models, though future work remains in areas like optimal partitioning and the automatic identification of fragment roles.

AI USE STATEMENT

We used generative AI tools to assist with translating parts of the paper into English. Generative AI tools were not used for model training or experimental evaluation. All authors reviewed the translated text, approved the final paper, and take responsibility for its content.

ETHICS STATEMENT

This work uses publicly available datasets and models and raises no ethical concerns specific to the proposed retrieval method.

REPRODUCIBILITY STATEMENT

Our research is designed for full reproducibility. All datasets, models, and training methodologies are described in detail to allow for easy replication. The experimental setup is thoroughly documented in Section 4.1, with specific training details, including hyperparameters and computational resources used, provided in Appendix C. We also provide a complete list of benchmark datasets used for evaluation in Appendix D.

REFERENCES

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, Mir Rosenberg, Xia Song, Alina Stoica, Saurabh Tiwary, and Tong Wang. Ms marco: A human generated machine reading comprehension dataset, 2016.

Chanyeol Choi, Junseong Kim, Seolhwa Lee, Jihoon Kwon, Sangmo Gu, Yejin Kim, Minkyung Cho, and Jy yong Sohn. Linq-embed-mistral technical report, 2024.

Gabriel de Souza P. Moreira, Radek Osmulski, Mengyao Xu, Ronay Ak, Benedikt Schifferer, and Even Oldridge. Nv-retriever: Improving text embedding models with effective hard-negative mining, 2024.

Daryna Dementieva, Nikolay Babakov, Amit Ronen, Abinew Ali Ayele, Naquee Rizwan, Florian Schneider, Xintong Wang, Seid Muhie Yimam, Daniil Alekhseevich Moskovskiy, Elisei Stakovskii, Eran Kaufman, Ashraf Elnagar, Animesh Mukherjee, and Alexander Panchenko. Multilingual and explainable text detoxification with parallel corpora. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert (eds.), *Proceedings of the 31st International Conference on Computational Linguistics*, pp. 7998–8025, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.535/.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL https://arxiv.org/abs/1810.04805.

Matthijs Douze, Alexandr Guzhva, Chengqi Deng, Jeff Johnson, Gergely Szilvasy, Pierre-Emmanuel Mazaré, Maria Lomeli, Lucas Hosseini, and Hervé Jégou. The faiss library. *IEEE Transactions on Big Data*, 2025.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. Eli5: Long form question answering, 2019.

Luyu Gao, Yunyi Zhang, Jiawei Han, and Jamie Callan. Scaling deep contrastive learning batch size under memory limited setup, 2021. URL https://arxiv.org/abs/2101.06983.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. Retrieval-augmented generation for large language models: A survey, 2023.

Michael Günther, Jackmin Ong, Isabelle Mohr, Alaeddine Abdessalem, Tanguy Abel, Mohammad Kalim Akram, Susana Guzman, Georgios Mastrapas, Saba Sturua, Bo Wang, Maximilian Werk, Nan Wang, and Han Xiao. Jina embeddings 2: 8192-token general-purpose text embeddings for long documents, 2024. URL https://arxiv.org/abs/2310.19923.

Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun hsuan Sung, Laszlo Lukacs, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply, 2017.

Xinshuo Hu, Zifei Shan, Xinping Zhao, Zetian Sun, Zhenyu Liu, Dongfang Li, Shaolin Ye, Xinyuan Wei, Qian Chen, Baotian Hu, Haofen Wang, Jun Yu, and Min Zhang. Kalm-embedding: Superior training data brings a stronger embedding model, 2025.

Zhengbao Jiang, Frank Xu, Luyu Gao, Zhiqing Sun, Qian Liu, Jane Dwivedi-Yu, Yiming Yang, Jamie Callan, and Graham Neubig. Active retrieval augmented generation. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 7969–7992, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.495. URL https://aclanthology.org/2023.emnlp-main.495/.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In Regina Barzilay and Min-Yen Kan (eds.), *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1147. URL https://aclanthology.org/P17-1147/.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Daniel Khashabi, Amos Ng, Tushar Khot, Ashish Sabharwal, Hannaneh Hajishirzi, and Chris Callison-Burch. GooAQ: Open question answering with diverse answer types. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 421–433, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.38. URL https://aclanthology.org/2021.findings-emnlp.38/.

Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert, 2020.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning, 2022.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL https://aclanthology.org/Q19-1026/.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. Nv-embed: Improved techniques for training llms as generalist embedding models, 2025a. URL https://arxiv.org/abs/2405.17428.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, Xiaoqi Ren, Shanfeng Zhang, Daniel Salz, Michael Boratko, Jay Han, Blair Chen, Shuo Huang, Vikram Rao, Paul Suganthan, Feng Han, Andreas Doumanoglou, Nithi Gupta, Fedor Moiseev, Cathy Yip, Aashi

Jain, Simon Baumgartner, Shahrokh Shahi, Frank Palma Gomez, Sandeep Mariserla, Min Choi, Parashar Shah, Sonam Goenka, Ke Chen, Ye Xia, Koert Chen, Sai Meher Karthik Duddu, Yichang Chen, Trevor Walker, Wenlei Zhou, Rakesh Ghiya, Zach Gleicher, Karan Gill, Zhe Dong, Mojtaba Seyedhosseini, Yunhsuan Sung, Raphael Hoffmann, and Tom Duerig. Gemini embedding: Generalizable embeddings from gemini, 2025b.

Yibin Lei, Tao Shen, Yu Cao, and Andrew Yates. Enhancing lexicon-based text embeddings with large language models. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 18986–19001, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.930. URL https://aclanthology.org/2025.acl-long.930/.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2020.

Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them, 2021.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models, 2020.

Chaofan Li, MingHao Qin, Shitao Xiao, Jianlyu Chen, Kun Luo, Yingxia Shao, Defu Lian, and Zheng Liu. Making text embedders few-shot learners, 2024.

Xianming Li and Jing Li. Aoe: Angle-optimized embeddings for semantic textual similarity. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 1825–1839, 2024.

Yuxin Liang, Rui Cao, Jie Zheng, Jie Ren, and Ling Gao. Learning to remove: Towards isotropic pre-trained bert embedding, 2021.

Ailiang Lin, Zhuoyun Li, Kotaro Funakoshi, and Manabu Okumura. Causal2vec: Improving decoder-only llms as versatile embedding models, 2025.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019. URL https://arxiv.org/abs/1907.11692.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

Macedo Maia, Siegfried Handschuh, André Freitas, Brian Davis, Ross McDermott, Manel Zarrouk, and Alexandra Balahur. Www'18 open challenge: Financial opinion mining and question answering. In *Companion Proceedings of the The Web Conference 2018*, WWW '18, pp. 1941–1942, Republic and Canton of Geneva, CHE, 2018. International World Wide Web Conferences Steering Committee. ISBN 9781450356404. doi: 10.1145/3184558.3192301. URL https://doi.org/10.1145/3184558.3192301.

Niklas Muennighoff, Nouamane Tazi, Loïc Magne, and Nils Reimers. Mteb: Massive text embedding benchmark, 2023. URL https://arxiv.org/abs/2210.07316.

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

Zach Nussbaum, John X. Morris, Brandon Duderstadt, and Andriy Mulyar. Nomic embed: Training a reproducible long context text embedder, 2025. URL https://arxiv.org/abs/2402.01613.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1264. URL https://aclanthology.org/D16-1264/.

Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks, 2019.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Bernard W Silverman. *Density estimation for statistics and data analysis*. Routledge, 2018.

Aditya Sinha, Siqi Zeng, Makoto Yamada, and Han Zhao. Learning structured representations with hyperbolic embeddings, 2024.

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

Harald Steck, Chaitanya Ekanadham, and Nathan Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM Web Conference 2024*, pp. 887–890, 2024.

Sotaro Takeshita, Yurina Takeshita, Daniel Ruffinelli, and Simone Paolo Ponzetto. Randomly removing 50% of dimensions in text embeddings has minimal impact on retrieval and classification tasks, 2025.

Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models, 2021.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In Marilyn Walker, Heng Ji, and Amanda Stent (eds.), *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1074. URL https://aclanthology.org/N18-1074/.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2018.

Henning Wachsmuth, Shahbaz Syed, and Benno Stein. Retrieval of the best counterargument without prior topic knowledge. In Iryna Gurevych and Yusuke Miyao (eds.), *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 241–251, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1023. URL https://aclanthology.org/P18-1023/.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7534–7550, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.609. URL https://aclanthology.org/2020.emnlp-main.609/.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere, 2020.

Benjamin Warner, Antoine Chaffin, Benjamin Clavié, Orion Weller, Oskar Hallström, Said Taghadouini, Alexis Gallagher, Raja Biswas, Faisal Ladhak, Tom Aarsen, Griffin Thomas Adams, Jeremy Howard, and Iacopo Poli. Smarter, better, faster, longer: A modern bidirectional encoder for fast, memory efficient, and long context finetuning and inference. In

Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 2526–2547, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.127. URL https://aclanthology.org/2025.acl-long.127/.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval, 2020.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering, 2018.

Kisung You. Semantics at an angle: When cosine similarity works until it doesn't, 2025.

Xin Zhang, Yanzhao Zhang, Dingkun Long, Wen Xie, Ziqi Dai, Jialong Tang, Huan Lin, Baosong Yang, Pengjun Xie, Fei Huang, Meishan Zhang, Wenjie Li, and Min Zhang. mgte: Generalized long-context text representation and reranking models for multilingual text retrieval, 2024. URL https://arxiv.org/abs/2407.19669.

Xinyu Zhang, Xueguang Ma, Peng Shi, and Jimmy Lin. Mr. TyDi: A multi-lingual benchmark for dense retrieval. In Duygu Ataman, Alexandra Birch, Alexis Conneau, Orhan Firat, Sebastian Ruder, and Gozde Gul Sahin (eds.), *Proceedings of the 1st Workshop on Multilingual Representation Learning*, pp. 127–137, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.mrl-1.12. URL https://aclanthology.org/2021.mrl-1.12/.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. MIRACL: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131, 2023a. doi: 10.1162/tacl_a_00595. URL https://aclanthology.org/2023.tacl-1.63/.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. MIRACL: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131, 2023b. doi: 10.1162/tacl_a_00595. URL https://aclanthology.org/2023.tacl-1.63/.

Yanzhao Zhang, Mingxin Li, Dingkun Long, Xin Zhang, Huan Lin, Baosong Yang, Pengjun Xie, An Yang, Dayiheng Liu, Junyang Lin, Fei Huang, and Jingren Zhou. Qwen3 embedding: Advancing text embedding and reranking through foundation models, 2025.

Yuqi Zhou, Sunhao Dai, Zhanshuo Cao, Xiao Zhang, and Jun Xu. Length-induced embedding collapse in plm-based models, 2024.

A LIMITATIONS

While the proposed Semantic Fragment Similarity (SFS) improves retrieval performance, several limitations highlight important directions for future work. First, although we have experimentally confirmed that SFS training induces a semantic division of labor among fragments, intuitively interpreting and naming the specific meaning that each fragment specializes in remains a challenge. The absence of a systematic method to quantify the semantic function of each fragment is a factor that limits the model's interpretability. This limitation in interpretability is also directly linked to another assumption of our study: that SFS computes the final score by simply averaging the similarity scores of the fragments. This approach carries the implicit premise that all fragments contribute equally to relevance judgments for any given task. If future research enables the identification and quantification of each fragment's semantic function, it would be possible to devise a more sophisticated aggregation mechanism that moves beyond simple averaging to dynamically assign weights tailored to the specific characteristics of a task.

Furthermore, determining the optimal fragment size currently relies on empirical hyperparameter tuning. While our extensive evaluations demonstrate consistent performance gains across various sizes, the ideal configuration may vary depending on the specific model architecture, embedding dimensionality, and the downstream task. The absence of a theoretically grounded or adaptive method to automatically determine the optimal partition limits the plug-and-play applicability of our approach. Future work should explore adaptive fragmentation strategies, such as learning dynamic fragment boundaries or determining context-aware fragment sizes during training, to further enhance both the robustness and usability of the model.

B EFFICIENT INFERENCE AND COMPATIBILITY WITH ANN SEARCH

A potential concern with the proposed Semantic Fragment Similarity (SFS) is the computational overhead associated with calculating cosine similarities for multiple fragments and aggregating them. In this section, we show that SFS can be reduced to a standard dot product operation via a simple vector transformation. This equivalence ensures that our method is fully compatible with existing Approximate Nearest Neighbor (ANN) search libraries (e.g., FAISS, ScaNN) without modifying the encoder or the ANN index.

B.1 COMPUTATIONAL COMPLEXITY ANALYSIS

In this section, we provide a theoretical analysis of the computational complexity of SFS compared to the standard global cosine similarity. For a concrete comparison, we adopt the configuration used in our main experiments: a model dimension of d_{model}=768 and a fragment size of d_{frag}=16, which partitions the vector into N_f=48 fragments.

To ensure a fair comparison of retrieval latency, we premise that all embedding vectors are **pre-normalized** during the offline indexing phase. Specifically, the standard approach assumes global L_2-normalization, while SFS assumes independent L_2-normalization for each of the N_f fragments. Under this precondition, the online similarity computation for both methods is reduced to dot product operations and accumulation.

Table 7: Theoretical FLOPs comparison for **vector normalization** (per vector) during the offline indexing phase. Although SFS increases the number of square root operations, the reduction in addition operations results in a similar overall operation count and identical asymptotic preprocessing complexity.

Method	Square	Add	Sqrt	Div	Total FLOPs
Global (1×768)	768	767	1	768	2,304
SFS (48×16)	768	720	48	768	2,304

For the standard global approach, computing the dot product of two 768-dimensional vectors requires 768 multiplications and 767 additions, resulting in a total of **1,535 floating-point operations (FLOPs)**. In contrast, the SFS calculation proceeds in three stages. First, computing the local dot products for $N_f = 48$ fragments requires $48 \times (16 \text{ multiplications} + 15 \text{ additions})$, totaling 1,488 FLOPs.

Table 8: Comparison of retrieval performance and computational efficiency on the Nano versions of MTEB datasets. All experiments utilized the FAISS `IndexHNSWFlat` on a single NVIDIA A6000 GPU.

Dataset	Configuration	Index Type	nDCG@10	Latency (ms)	QPS
SciFact	Global (1×768)	HNSWFlat	0.6126	**0.1318**	**7587.93**
	SFS (48×16)	HNSWFlat	**0.6614**	0.2401	4164.07
NFCorpus	Global (1×768)	HNSWFlat	0.2114	**0.1449**	**6901.93**
	SFS (48×16)	HNSWFlat	**0.2520**	0.1486	6730.05
ClimateFEVER	Global (1×768)	HNSWFlat	0.2792	**0.1548**	**6458.34**
	SFS (48×16)	HNSWFlat	**0.3097**	0.2145	4662.41

Second, aggregating these partial scores involves $48 - 1 = 47$ additions. Finally, a single division by N_f is performed to compute the average similarity. This sums to **1,536 FLOPs** $(1,488 + 47 + 1)$. Comparing the two methods, the difference amounts to a small constant-factor change (i.e., the final division by N_f). Therefore, SFS maintains the same $\mathcal{O}(d_{\text{model}})$ asymptotic complexity as the standard operation.

B.2 Mathematical Equivalence to Inner Product

SFS score $S_{frag}(q, d)$ is defined as the average of cosine similarities between corresponding fragments of the query embedding E_q and document embedding E_d:

$$\begin{aligned} S_{frag}(q,d) &= \frac{1}{N_f}\sum_{i=1}^{N_f}\cos(f_{qi}, f_{di}) \\ &= \frac{1}{N_f}\sum_{i=1}^{N_f}\frac{f_{qi}\cdot f_{di}}{\|f_{qi}\|\|f_{di}\|} \end{aligned} \tag{5}$$

where f_{qi} and f_{di} denote the i-th fragments of the query and document vectors, respectively, and N_f represents the total number of fragments (i.e., $N_f = d_{model}/d_{frag}$). We introduce a transformation function $\phi(\cdot)$ that applies L_2-normalization to each fragment individually and concatenates them into a single vector:

$$\phi(E) = \left[\frac{f_1}{\|f_1\|} \oplus \frac{f_2}{\|f_2\|} \oplus \cdots \oplus \frac{f_{N_f}}{\|f_{N_f}\|}\right] \in \mathbb{R}^{d_{model}} \tag{6}$$

Here, $\oplus$ denotes the vector concatenation operation. The dimension of the transformed vector $\phi(E)$ remains identical to the original embedding dimension d_{model}. By substituting the normalized fragments into the original equation, the SFS score can be rewritten as a standard dot product:

$$\begin{aligned} \phi(E_q)\cdot\phi(E_d) &= \sum_{i=1}^{N_f}\left(\frac{f_{qi}}{\|f_{qi}\|}\cdot\frac{f_{di}}{\|f_{di}\|}\right) \\ &= \sum_{i=1}^{N_f}\cos(f_{qi}, f_{di}) \\ &= N_f \times S_{frag}(q,d) \end{aligned} \tag{7}$$

Since N_f is a constant hyperparameter, maximizing the standard dot product $\phi(E_q)\cdot\phi(E_d)$ is equivalent to maximizing the SFS score $S_{frag}(q, d)$.

B.3 Compatibility with ANN Search Frameworks

To demonstrate the practical scalability of Semantic Fragment Similarity (SFS), we evaluated its performance using the widely adopted FAISS (Douze et al., 2025) library[1], with a ModernBERT

[1] `https://github.com/facebookresearch/faiss`

model. The primary objective was to verify that SFS can be seamlessly integrated into standard Approximate Nearest Neighbor (ANN) search pipelines without requiring architectural modifications.

Experiments were conducted on three datasets using the `IndexHNSWFlat` index on a single NVIDIA A6000 GPU. As detailed in Table 8, the SFS configuration (48×16) consistently achieves higher retrieval performance (nDCG@10) than the standard global vector approach (1×768). These results empirically validate that SFS is fully compatible with existing ANN infrastructure, confirming its practical compatibility in large-scale retrieval systems.

C Training Details

C.1 Setup

All training is conducted on four NVIDIA A6000 GPUs. We employ GradCache (Gao et al., 2021) to train with a large batch size (512) with limited GPU memory. Models are trained for 2 epochs with a maximum sequence length of 512, using bf16 precision. We use the AdamW optimizer (Loshchilov & Hutter, 2019) with a learning rate of 1e-4 and a linear learning rate warm-up for 5% of the total steps. Additionally, to ensure reproducibility, all random seeds are fixed.

Table 9: Instructions and number of samples used for each training dataset.

Task Name	Instruction	# of samples
arguana	Given a claim, retrieve documents that support or refute the claim	4.1k
fever	Given a claim, retrieve documents that support or refute the claim	29.1k
scifact	Given a scientific claim, retrieve documents that support or refute the claim	0.5k
paq	Given a web search query, retrieve relevant passages that answer the query	45.2k
msmarco_document	Given a question, retrieve documents that can help answer the question	50k
msmarco_passage	Given a question, retrieve passages that can help answer the question	50k
squad	Given a question, retrieve passages that answer the question	87.6k
natural question	Given a question, retrieve passages that answer the question	58.6k
hotpotqa	Given a multi-hop question, retrieve documents that can help answer the question	84.5k
fiqa	Given a financial question, retrieve relevant passages that answer the query	5.5k
miracl	Given a question, retrieve passages that answer the question	7.9k
mrtydi	Given a question, retrieve passages that answer the question	3.5k
gooaq	Given a question, retrieve passages that answer the question	40.4k
eli5	Given a question, retrieve passages that answer the question	21k
trivia_qa	Given a question, retrieve passages that answer the question	31.7k

C.2 Dataset

We leverage the training data provided by BGE-en-ICL (Li et al., 2024) as follows (Lei et al., 2025) along with a collection of publicly available retrieval datasets. We adopt the retrieval datasets as follows: ArguAna (Wachsmuth et al., 2018), FEVER (Thorne et al., 2018), SciFact (Wadden et al., 2020), PAQ (Lewis et al., 2021), MSMARCO (Bajaj et al., 2016), SQuAD (Rajpurkar et al., 2016), Natural Question (Kwiatkowski et al., 2019), HotpotQA (Yang et al., 2018), FiQA (Maia et al., 2018), MIRACL (Zhang et al., 2023b), Mr.TyDi (Zhang et al., 2021), GooAQ (Khashabi et al., 2021), ELI5 (Fan et al., 2019) and TriviaQA (Joshi et al., 2017). The full list of detailed datasets and their corresponding instructions is provided in Table 9.

Hard Negative Mining To construct a high-quality training dataset, we employ a hard negative mining process (de Souza P. Moreira et al., 2024). This process begins by utilizing bge-large-en-v1.5 (Xiao et al., 2023) to convert all queries and passages into dense vector representations. For each query, we perform an efficient, corpus-wide semantic search to rank all passages by similarity. From this ranked list, we select effective hard negatives. We adopt a top sampling strategy and apply an absolute margin of 0.04. From the resulting candidates, we sample 7 hard negatives for each sample.

Table 10: Subset of MTEB tasks and benchmarks used for our experiments.

Task Category	Benchmarks
Retrieval (12)	NanoArguAna, NanoClimateFEVER, NanoDBPEDIA, NanoFEVER, NanoFiQA2018, NanoHotpotQA, NanoMSMARCO, NanoNFCorpus, NanoQuoraRetrieval, NanoSCIDOCS, NanoSciFact, NanoTouche2020
Reranking (4)	AskUbuntuDupQuestions, MindSmallRerank, SciDocsRR, StackOverflowDupQuestions
Classification (7)	AmazonCounterfactual, AmazonPolarity, AmazonReviews, Banking77, Imdb, ToxicConversations, TweetSentimentExtraction

D Evaluation Details

Table 10 lists all the evaluation tasks. We evaluate all five fine-tuned models on the MTEB Benchmark encompassing 12 retrieval datasets, 4 reranking datasets and 7 classification datasets. Due to the significant computational cost of the full benchmark, we specifically utilize NanoBEIR (Thakur et al., 2021) for the retrieval tasks (Takeshita et al., 2025). For the classification, we include datasets that are evaluated using the k-NN classification. Furthermore, to align with our model's learning mechanism, similarity is specifically measured using our proposed fragmentation similarity. For evaluation, we use the prompts provided by the library [2] for each benchmark.

E Implementation Details for Flattening Analysis

To analyze the statistical properties of the representations, we utilize the embedding vectors encoded by the models.

E.1 Kernel Density Estimation (KDE)

We perform Kernel Density Estimation (KDE) (Silverman, 2018) to visualize and compare the overall distribution of the embedding coordinate values, specifically to assess the degree of flattening. To preserve the intrinsic distributional shape induced by the embeddings, we collect all coordinate values (a total of $N \times 768$) from the encoded embeddings without applying any normalization prior to estimating the density. For the KDE, we employ a Gaussian kernel, with the bandwidth determined by Silverman's rule of thumb (Silverman, 2018): $h = 1.06 \cdot \sigma \cdot n^{-1/5}$ (where σ denotes the empirical standard deviation of all coordinate values, and n represents the total number of coordinates). Furthermore, to mitigate scale distortion caused by extreme outliers in the visualizations, the x-axis range of the plots is restricted to the 0.5th to 99.5th percentile interval of the coordinate values, although the KDE itself is computed using the entire set of coordinates.

E.2 Entropy and Kurtosis Analysis

To quantify the representation flattening phenomenon at the vector level, we calculate the Shannon Entropy (Shannon, 1948) and Pearson Kurtosis for each sample embedding vector $v \in \mathbb{R}^{768}$. These metrics are then used to construct a scatter plot, where each point represents a single vector. Unlike the aforementioned KDE process, L2 normalization is applied to each sample vector prior to calculating these scatter plot metrics.

First, to quantify the degree to which the coordinate distribution of a vector is concentrated along specific dimensions, the normalized vector $\tilde{v}$ is mapped to a probability distribution p via a temperature-scaled softmax function:

$$p_j = \frac{\exp(\tilde{v}_j/T)}{\sum_{k=1}^{768} \exp(\tilde{v}_k/T)}, \quad j = 1, \ldots, 768.$$

[2] https://github.com/embeddings-benchmark/mteb

Based on this, the Shannon entropy $H(\tilde{v}; T) = -\sum_{j=1}^{768} p_j \log p_j$ is computed using the natural logarithm. The softmax operation is employed not to assign a probabilistic interpretation, but to map the coordinate values onto a simplex space, enabling us to diagnose the peakiness and flatness of the distribution. For this computation, the temperature parameter is fixed at $T = 0.1$.

Second, to measure the tail heaviness and peakedness of the coordinate distribution, the Pearson kurtosis $\kappa(\tilde{v})$ is calculated for the same normalized vector $\tilde{v}$:

$$\kappa(\tilde{v}) = \frac{E[(\tilde{v} - \mu)^4]}{(E[(\tilde{v} - \mu)^2])^2}$$

Here, μ represents the mean of the 768 coordinates of $\tilde{v}$, and $E[\cdot]$ denotes the expected value (empirical mean) across the coordinate dimensions.

F Further Experimental Results

F.1 Fragment granularity scaling

Setup To more thoroughly investigate the impact of fragment granularity, we evaluate an expanded range of fragment configurations on BERT and ModernBERT. In addition to those previously evaluated, we include intermediate granularities ($d_{\text{frag}} \in \{384, 256, 192, 128\}$) and more fine-grained settings ($d_{\text{frag}} \in \{8, 4, 2\}$).

Table 11: The effect of fragment granularity scaling on retrieval performance.

Model	(1,768)	(2,384)	(3,256)	(4,192)	(6,128)	(12,64)	(24,32)	(48,16)	(96,8)	(192,4)	(384,2)
BERT	0.5178	0.5191	0.5136	0.5175	0.5185	0.5163	**0.5197**	0.5181	<u>0.5192</u>	0.5144	0.5053
ModernBERT	0.5375	0.5370	0.5345	0.5368	0.5330	0.5257	0.5483	**0.5512**	<u>0.5502</u>	0.5375	0.5242

Results Table 11 reveals a clear trend with respect to fragment granularity. A consistent pattern is observed where performance improves as fragments become more granular, peaks at a certain point, and then sharply declines when the fragmentation becomes extreme. Generally, optimal performance is achieved within the more granular range of fragment sizes, specifically around 32, 16, and 8. This pattern suggests a fundamental trade-off related to the role of fragments. Fragments must be small enough to encode distinct semantic aspects, yet large enough to retain the minimum dimensional capacity required to encode complex meanings. Extreme fragmentation undermines this capacity, leading to performance degradation. Therefore, These findings suggest that an optimal level of granularity exists to maximize the effectiveness of SFS.

F.2 Detailed Results

We provide a comprehensive breakdown of the performance metrics for each model and component analysis evaluated in our study.

First, we present the detailed results for the main experiments. The performance breakdown for `ModernBERT` is presented in Table 15, followed by the results for `BERT` in Table 12, `GTE-en-MLM` in Table 16, `NomicBERT` in Table 14, and `RoBERTa` in Table 13.

Additionally, we provide the results from our ablation and component analysis studies. Table 17 details the impact of different **pooling methods**, while the effects of **representation scaling** are shown in Table 18. The results related to the **decoder architecture** and the choice of **similarity metric** are documented in Table 19 and Table 20, respectively. Finally, the analysis of the **Matryoshka Representation Learning (MRL)** training objective is provided in Table 21.

Table 12: Full Benchmark Results for BERT

Dataset	(1,768)	(3,256)	(6,128)	(12,64)	(24,32)	(48,16)
ArguAna	0.5780	0.5757	0.5873	0.5837	0.5963	0.5736
ClimateFEVER	0.2224	0.2273	0.2290	0.2243	0.2266	0.2181
DBPEDIA	0.5148	0.5129	0.5132	0.5089	0.5168	0.5155
FEVER	0.8221	0.8034	0.8208	0.8172	0.8253	0.8306
FiQA2018	0.3847	0.3730	0.3810	0.3862	0.3788	0.3774
HotpotQA	0.6313	0.6336	0.6364	0.6222	0.6263	0.6330
MSMARCO	0.5693	0.5595	0.5609	0.5503	0.5721	0.5661
NFCorpus	0.2542	0.2627	0.2628	0.2679	0.2646	0.2690
QuoraRetrieval	0.9267	0.9208	0.9210	0.9199	0.9193	0.9203
SCIDOCS	0.2475	0.2446	0.2437	0.2468	0.2458	0.2487
SciFact	0.5675	0.5644	0.5685	0.5648	0.5651	0.5571
Touche2020	0.4695	0.4615	0.4688	0.4665	0.4679	0.4739
AskUbuntuDupQuestions	0.5502	0.5551	0.5533	0.5534	0.5585	0.5570
SciDocsRR	0.7021	0.7021	0.7023	0.7017	0.7037	0.7046
StackOverflowDupQuestions	0.4248	0.4266	0.4284	0.4271	0.4279	0.4302
MindSmallRerank	0.3117	0.3117	0.3113	0.3113	0.3113	0.3111
AmazonCounterfactual	0.5509	0.5535	0.5569	0.5679	0.5773	0.5730
AmazonPolarity	0.5981	0.5971	0.5969	0.6002	0.6021	0.6034
AmazonReviews	0.2756	0.2751	0.2745	0.2742	0.2773	0.2777
Banking77	0.6849	0.6859	0.6884	0.6869	0.6900	0.6902
Imdb	0.5725	0.5725	0.5707	0.5692	0.5761	0.5772
ToxicConversations	0.5027	0.5048	0.5090	0.5326	0.5099	0.5106
TweetSentimentExtraction	0.4285	0.4320	0.4293	0.4281	0.4316	0.4313

Table 13: Full Benchmark Results for RoBERTa

Dataset	(1,768)	(3,256)	(6,128)	(12,64)	(24,32)	(48,16)
ArguAna	0.6112	0.6030	0.6194	0.6198	0.6258	0.6256
ClimateFEVER	0.2098	0.2481	0.2348	0.2414	0.2225	0.2549
DBPEDIA	0.4890	0.4857	0.4929	0.4983	0.5017	0.5011
FEVER	0.6952	0.6969	0.6939	0.6800	0.6886	0.7053
FiQA2018	0.4101	0.4284	0.4317	0.4432	0.4304	0.4436
HotpotQA	0.6102	0.6178	0.6209	0.6288	0.6540	0.6392
MSMARCO	0.5601	0.5590	0.5513	0.5835	0.5763	0.5569
NFCorpus	0.2357	0.2373	0.2292	0.2397	0.2380	0.2386
QuoraRetrieval	0.9415	0.9447	0.9361	0.9433	0.9260	0.9462
SCIDOCS	0.2720	0.2768	0.2797	0.2857	0.2774	0.2916
SciFact	0.5163	0.5001	0.4999	0.5263	0.5436	0.5147
Touche2020	0.5126	0.5079	0.5110	0.5118	0.5118	0.5258
AskUbuntuDupQuestions	0.5605	0.5591	0.5622	0.5560	0.5611	0.5582
MindSmallRerank	0.3213	0.3202	0.3198	0.3208	0.3202	0.3182
SciDocsRR	0.6956	0.6964	0.6926	0.6967	0.6977	0.6977
StackOverflowDupQuestions	0.4176	0.4185	0.4163	0.4165	0.4194	0.4259
AmazonCounterfactual	0.5427	0.5385	0.5388	0.5981	0.6191	0.6002
AmazonPolarity	0.6072	0.5991	0.6030	0.6004	0.6048	0.5931
AmazonReviews	0.2787	0.2761	0.2841	0.2809	0.2845	0.2823
Banking77	0.7237	0.7199	0.7232	0.7207	0.7240	0.7267
Imdb	0.5937	0.5932	0.5939	0.5757	0.5820	0.5762
ToxicConversations	0.5371	0.5611	0.5723	0.6117	0.5885	0.5979
TweetSentimentExtraction	0.4266	0.4176	0.4293	0.4344	0.4297	0.4238

Table 14: Full Benchmark Results for NomicBERT

Dataset	(1,768)	(3,256)	(6,128)	(12,64)	(24,32)	(48,16)
ArguAna	0.5795	0.5685	0.5944	0.6109	0.5997	0.5974
ClimateFEVER	0.3034	0.2915	0.2894	0.2947	0.2923	0.3059
DBPEDIA	0.5305	0.5398	0.5480	0.5399	0.5334	0.5482
FEVER	0.7814	0.8017	0.8181	0.8000	0.8120	0.8068
FiQA2018	0.4463	0.4573	0.4556	0.4580	0.4472	0.4403
HotpotQA	0.6766	0.6631	0.6628	0.6741	0.6950	0.6950
MSMARCO	0.5748	0.5860	0.5840	0.6115	0.5863	0.5925
NFCorpus	0.2303	0.2229	0.2192	0.2219	0.2291	0.2329
QuoraRetrieval	0.8992	0.9095	0.9258	0.9143	0.9212	0.9342
SCIDOCS	0.2724	0.2630	0.2507	0.2638	0.2690	0.2657
SciFact	0.5880	0.5865	0.5958	0.5881	0.5876	0.5741
Touche2020	0.5429	0.5251	0.5282	0.5286	0.5253	0.5299
AskUbuntuDupQuestions	0.5505	0.5525	0.5501	0.5521	0.5508	0.5516
MindSmallRerank	0.3052	0.3041	0.3062	0.3055	0.3053	0.3048
SciDocsRR	0.7085	0.7070	0.7068	0.7085	0.7114	0.7126
StackOverflowDupQuestions	0.4246	0.4262	0.4168	0.4181	0.4169	0.4188
AmazonCounterfactual	0.4996	0.5010	0.5043	0.5134	0.5231	0.5105
AmazonPolarity	0.5834	0.5857	0.5808	0.5899	0.5932	0.5934
AmazonReviews	0.2683	0.2682	0.2702	0.2733	0.2737	0.2740
Banking77	0.7002	0.6996	0.7009	0.7046	0.7072	0.7063
Imdb	0.5579	0.5587	0.5630	0.5618	0.5622	0.5610
ToxicConversations	0.4737	0.4845	0.4879	0.5224	0.5269	0.5282
TweetSentimentExtraction	0.4203	0.4184	0.4188	0.4223	0.4148	0.4207

Table 15: Full Benchmark Results for ModernBERT

Dataset	(1,768)	(3,256)	(6,128)	(12,64)	(24,32)	(48,16)
ArguAna	0.5932	0.5979	0.5802	0.5852	0.5950	0.5937
ClimateFEVER	0.2782	0.2682	0.2567	0.1862	0.2969	0.2994
DBPEDIA	0.4932	0.5012	0.4954	0.5143	0.5108	0.5100
FEVER	0.7746	0.7867	0.8071	0.7306	0.7968	0.8051
FiQA2018	0.4657	0.4711	0.4703	0.4734	0.4851	0.4726
HotpotQA	0.6561	0.6489	0.6505	0.6582	0.6612	0.6776
MSMARCO	0.5797	0.5680	0.5803	0.5713	0.6027	0.5755
NFCorpus	0.2263	0.2325	0.2248	0.2208	0.2387	0.2544
QuoraRetrieval	0.9450	0.9521	0.9405	0.9494	0.9552	0.9408
SCIDOCS	0.2979	0.3027	0.2982	0.2941	0.2849	0.2911
SciFact	0.6228	0.6040	0.6041	0.5936	0.6558	0.6640
Touche2020	0.5279	0.5081	0.5179	0.5114	0.5146	0.5201
AskUbuntuDupQuestions	0.5895	0.5909	0.5891	0.5855	0.5849	0.5892
MindSmallRerank	0.3100	0.3133	0.3122	0.3171	0.3143	0.3154
SciDocsRR	0.7290	0.7320	0.7323	0.7319	0.7396	0.7406
StackOverflowDupQuestions	0.4756	0.4712	0.4733	0.4692	0.4746	0.4760
AmazonCounterfactual	0.5296	0.5333	0.5371	0.5468	0.5694	0.5686
AmazonPolarity	0.5875	0.5883	0.5883	0.5785	0.5895	0.5970
AmazonReviews	0.2775	0.2717	0.2719	0.2675	0.2765	0.2803
Banking77	0.7587	0.7471	0.7504	0.7501	0.7509	0.7537
Imdb	0.5822	0.5863	0.5832	0.5754	0.5951	0.5953
ToxicConversations	0.4737	0.4807	0.4809	0.5165	0.5126	0.5281
TweetSentimentExtraction	0.4090	0.4094	0.4051	0.4156	0.4168	0.4148

Table 16: Full Benchmark Results for GTE-en-MLM

Dataset	(1,768)	(3,256)	(6,128)	(12,64)	(24,32)	(48,16)
ArguAna	0.5958	0.5943	0.5989	0.6015	0.6040	0.5975
ClimateFEVER	0.2631	0.2710	0.2408	0.2756	0.2802	0.2846
DBPEDIA	0.5206	0.5248	0.5260	0.5127	0.5143	0.5271
FEVER	0.7952	0.7885	0.7445	0.7837	0.7900	0.7809
FiQA2018	0.4386	0.4685	0.4699	0.4812	0.4835	0.4799
HotpotQA	0.6389	0.6516	0.6440	0.6606	0.6618	0.6655
MSMARCO	0.6039	0.5871	0.6129	0.6220	0.6177	0.6099
NFCorpus	0.2884	0.2802	0.2866	0.2871	0.2839	0.2834
QuoraRetrieval	0.9236	0.9172	0.9134	0.9214	0.9197	0.9264
SCIDOCS	0.2842	0.2893	0.2915	0.2996	0.2949	0.3023
SciFact	0.6226	0.6088	0.6150	0.6126	0.6284	0.6077
Touche2020	0.5180	0.5252	0.5250	0.5322	0.5379	0.5361
AskUbuntuDupQuestions	0.5825	0.5790	0.5771	0.5762	0.5773	0.5771
MindSmallRerank	0.3155	0.3162	0.3172	0.3167	0.3169	0.3156
SciDocsRR	0.7304	0.7321	0.7287	0.7314	0.7324	0.7321
StackOverflowDupQuestions	0.4409	0.4382	0.4367	0.4396	0.4446	0.4446
AmazonCounterfactual	0.5009	0.4949	0.4954	0.4982	0.5046	0.4828
AmazonPolarity	0.6007	0.6020	0.6029	0.6060	0.6083	0.5943
AmazonReviews	0.2774	0.2781	0.2755	0.2839	0.2860	0.2843
Banking77	0.7394	0.7404	0.7407	0.7453	0.7456	0.7445
Imdb	0.5663	0.5693	0.5743	0.5815	0.5801	0.5779
ToxicConversations	0.4664	0.4663	0.5046	0.5184	0.5174	0.5185
TweetSentimentExtraction	0.4277	0.4234	0.4223	0.4168	0.4121	0.4113

Table 17: Retrieval Benchmark Results for Average Pooling Methods

	ModernBERT					
Dataset	**(1,768)**	**(3,256)**	**(6,128)**	**(12,64)**	**(24,32)**	**(48,16)**
ArguAna	0.5976	0.6003	0.6136	0.6001	0.6095	0.5961
ClimateFEVER	0.2759	0.2945	0.2897	0.3080	0.3061	0.3152
DBPEDIA	0.5233	0.5202	0.5173	0.4930	0.5021	0.4912
FEVER	0.7949	0.7731	0.7847	0.7854	0.7887	0.7856
FiQA2018	0.4406	0.4471	0.4525	0.4455	0.4884	0.4886
HotpotQA	0.6555	0.6560	0.6401	0.6608	0.6638	0.6614
MSMARCO	0.5380	0.5266	0.5556	0.5255	0.5494	0.5469
NFCorpus	0.2502	0.2864	0.2658	0.2478	0.2666	0.2532
QuoraRetrieval	0.9518	0.9357	0.9361	0.9233	0.9550	0.9313
SCIDOCS	0.2922	0.2949	0.3060	0.2832	0.2958	0.2909
SciFact	0.6278	0.6556	0.6322	0.6352	0.6285	0.6534
Touche2020	0.5123	0.5036	0.5005	0.4953	0.5118	0.5147
	BERT					
Dataset	**(1,768)**	**(3,256)**	**(6,128)**	**(12,64)**	**(24,32)**	**(48,16)**
ArguAna	0.5951	0.5971	0.5899	0.6012	0.5939	0.6004
ClimateFEVER	0.2527	0.2536	0.2598	0.2637	0.2461	0.2470
DBPEDIA	0.5091	0.5162	0.5184	0.5248	0.5271	0.5136
FEVER	0.8118	0.8108	0.8219	0.8272	0.8276	0.8382
FiQA2018	0.3796	0.3703	0.3708	0.3797	0.3766	0.3653
HotpotQA	0.6386	0.6484	0.6424	0.6431	0.6428	0.6384
MSMARCO	0.5651	0.5783	0.5561	0.5531	0.5451	0.5662
NFCorpus	0.2572	0.2438	0.2495	0.2495	0.2473	0.2558
QuoraRetrieval	0.9303	0.9329	0.9230	0.9305	0.9306	0.9379
SCIDOCS	0.2481	0.2464	0.2479	0.2554	0.2537	0.2525
SciFact	0.5841	0.5858	0.5801	0.5802	0.5701	0.5732
Touche2020	0.4695	0.4693	0.4663	0.4618	0.4641	0.4662

Table 18: Retrieval Benchmark Results for Fragment Granularity Scaling

	ModernBERT										
Dataset	**(1, 768)**	**(2, 384)**	**(3, 256)**	**(4, 192)**	**(6, 128)**	**(12, 64)**	**(24, 32)**	**(48, 16)**	**(96, 8)**	**(192, 4)**	**(384, 2)**
ArguAna	0.5932	0.5853	0.5979	0.5890	0.5802	0.5852	0.5950	0.5937	0.6031	0.5885	0.5743
ClimateFEVER	0.2782	0.2617	0.2682	0.2608	0.2567	0.1862	0.2969	0.2994	0.3035	0.2920	0.2731
DBPEDIA	0.4932	0.4939	0.5012	0.4962	0.4954	0.5143	0.5108	0.5100	0.5081	0.5030	0.4900
FEVER	0.7746	0.8147	0.7867	0.7878	0.8071	0.7306	0.7968	0.8051	0.8079	0.7846	0.7619
FiQA2018	0.4657	0.4689	0.4711	0.4835	0.4703	0.4734	0.4851	0.4726	0.4789	0.4668	0.4400
HotpotQA	0.6561	0.6482	0.6489	0.6435	0.6505	0.6582	0.6612	0.6776	0.6700	0.6540	0.6188
MSMARCO	0.5797	0.5976	0.5680	0.6046	0.5803	0.5713	0.6027	0.5755	0.5702	0.5501	0.5400
NFCorpus	0.2263	0.2237	0.2325	0.2241	0.2248	0.2208	0.2387	0.2544	0.2391	0.2318	0.2298
QuoraRetrieval	0.9450	0.9562	0.9521	0.9544	0.9405	0.9494	0.9552	0.9408	0.9676	0.9544	0.9588
SCIDOCS	0.2979	0.2881	0.3027	0.2869	0.2982	0.2941	0.2849	0.2911	0.2848	0.2654	0.2578
SciFact	0.6228	0.6005	0.6040	0.6055	0.6041	0.5936	0.6558	0.6640	0.6570	0.6487	0.6064
Touche2020	0.5279	0.5175	0.5081	0.5243	0.5179	0.5114	0.5146	0.5201	0.5171	0.5183	0.5404
	BERT										
Dataset	**(1, 768)**	**(2, 384)**	**(3, 256)**	**(4, 192)**	**(6, 128)**	**(12, 64)**	**(24, 32)**	**(48, 16)**	**(96, 8)**	**(192, 4)**	**(384, 2)**
ArguAna	0.5780	0.5841	0.5757	0.5724	0.5873	0.5837	0.5963	0.5736	0.5717	0.5847	0.5490
ClimateFEVER	0.2224	0.2285	0.2273	0.2399	0.2290	0.2243	0.2266	0.2181	0.2221	0.2052	0.2285
DBPEDIA	0.5148	0.5108	0.5129	0.5142	0.5132	0.5089	0.5168	0.5155	0.5208	0.5177	0.5099
FEVER	0.8221	0.8133	0.8034	0.8161	0.8208	0.8172	0.8253	0.8306	0.8212	0.8009	0.8504
FiQA2018	0.3847	0.3879	0.3730	0.3725	0.3810	0.3862	0.3788	0.3774	0.3798	0.3640	0.3367
HotpotQA	0.6313	0.6462	0.6336	0.6474	0.6364	0.6222	0.6263	0.6330	0.6258	0.6210	0.6059
MSMARCO	0.5693	0.5703	0.5595	0.5685	0.5609	0.5503	0.5721	0.5661	0.5938	0.5709	0.5378
NFCorpus	0.2542	0.2652	0.2627	0.2610	0.2628	0.2679	0.2646	0.2690	0.2605	0.2610	0.2540
QuoraRetrieval	0.9267	0.9175	0.9208	0.9192	0.9210	0.9199	0.9193	0.9203	0.9206	0.9300	0.9293
SCIDOCS	0.2475	0.2427	0.2446	0.2362	0.2437	0.2468	0.2458	0.2487	0.2540	0.2409	0.2463
SciFact	0.5675	0.5740	0.5644	0.5722	0.5685	0.5648	0.5651	0.5571	0.5822	0.5692	0.4953
Touche2020	0.4695	0.4726	0.4615	0.4689	0.4688	0.4665	0.4679	0.4739	0.4638	0.4599	0.4729

Table 19: Retrieval Benchmark Results for Decoder Models

	Qwen3-0.6B							
Dataset	**(1, 2048)**	**(2, 1024)**	**(4, 512)**	**(8, 256)**	**(16, 128)**	**(32, 64)**	**(64, 32)**	**(128, 16)**
ArguAna	0.5885	0.6229	0.6168	0.6039	0.6081	0.6289	0.6214	0.6221
ClimateFEVER	0.3585	0.3471	0.3592	0.3488	0.3341	0.3686	0.3524	0.3148
DBPEDIA	0.5104	0.5084	0.5371	0.5004	0.5477	0.5416	0.5305	0.5524
FEVER	0.8833	0.8796	0.8293	0.8745	0.8398	0.8627	0.8586	0.8196
FiQA2018	0.4891	0.4889	0.5152	0.4904	0.5180	0.5193	0.5160	0.4853
HotpotQA	0.7246	0.7322	0.7216	0.7176	0.7384	0.7088	0.7413	0.7248
MSMARCO	0.5260	0.5643	0.6162	0.5666	0.5790	0.5945	0.5784	0.6013
NFCorpus	0.2719	0.2652	0.2897	0.2881	0.2898	0.2798	0.2855	0.3017
QuoraRetrieval	0.9414	0.9409	0.9397	0.9405	0.9396	0.9429	0.9450	0.9470
SCIDOCS	0.3770	0.3717	0.3735	0.3801	0.3774	0.3803	0.3701	0.3829
SciFact	0.7464	0.7441	0.7366	0.7251	0.7333	0.7511	0.7520	0.7659
Touche2020	0.5146	0.5031	0.5276	0.5375	0.5066	0.5159	0.5208	0.5410
	Llama-3.2-1B							
Dataset	**(1, 2048)**	**(2, 1024)**	**(4, 512)**	**(8, 256)**	**(16, 128)**	**(32, 64)**	**(64, 32)**	**(128, 16)**
ArguAna	0.6451	0.6215	0.6365	0.6276	0.6407	0.6349	0.6320	0.6546
ClimateFEVER	0.3129	0.3086	0.3043	0.3208	0.3142	0.3194	0.3278	0.3183
DBPEDIA	0.5191	0.5514	0.5425	0.5589	0.5550	0.5344	0.5445	0.5590
FEVER	0.8917	0.9214	0.8981	0.9095	0.8804	0.8720	0.8583	0.8919
FiQA2018	0.5454	0.5374	0.5368	0.5391	0.5450	0.5205	0.5448	0.5165
HotpotQA	0.7963	0.7843	0.7740	0.7764	0.7683	0.7944	0.7817	0.7810
MSMARCO	0.5589	0.5831	0.5961	0.5858	0.5965	0.6122	0.6289	0.5985
NFCorpus	0.3253	0.3463	0.3298	0.3338	0.3180	0.3345	0.3350	0.3354
QuoraRetrieval	0.9387	0.9483	0.9470	0.9505	0.9506	0.9351	0.9433	0.9441
SCIDOCS	0.3712	0.3592	0.3709	0.3719	0.3646	0.3642	0.3554	0.3672
SciFact	0.7324	0.7422	0.7563	0.7333	0.7380	0.7287	0.7243	0.7532
Touche2020	0.4973	0.5170	0.5093	0.5131	0.5089	0.5065	0.5151	0.5076

Table 20: Retrieval Benchmark Results for ModernBERT with Different Similarity Metrics

Dataset	**Cosine**		**Euclidean**		**Manhattan**		**Dot**	
	(1, 768)	**(48, 16)**	**(1, 768)**	**(48, 16)**	**(1, 768)**	**(48, 16)**	**(1, 768)**	**(48, 16)**
ArguAna	0.5932	0.5937	0.4801	0.5434	0.5060	0.5886	0.4800	0.5597
ClimateFEVER	0.2782	0.2994	0.0492	0.2157	0.0465	0.2150	0.1143	0.2419
DBPEDIA	0.4932	0.5100	0.2358	0.3313	0.2963	0.5163	0.2293	0.4879
FEVER	0.7746	0.8051	0.0885	0.7061	0.1142	0.7159	0.5152	0.7304
FiQA2018	0.4657	0.4726	0.2764	0.3646	0.2997	0.4548	0.2037	0.4403
HotpotQA	0.6561	0.6776	0.6312	0.6576	0.6316	0.6218	0.6847	0.7034
MSMARCO	0.5797	0.5755	0.3738	0.4660	0.4001	0.5457	0.2653	0.5287
NFCorpus	0.2263	0.2544	0.1179	0.1647	0.1464	0.2030	0.1135	0.2014
QuoraRetrieval	0.9450	0.9408	0.9259	0.9338	0.9288	0.9475	0.8081	0.9536
SCIDOCS	0.2979	0.2911	0.2080	0.2374	0.2667	0.2619	0.1503	0.2706
SciFact	0.6228	0.6640	0.4414	0.5955	0.4716	0.5986	0.4076	0.6039
Touche2020	0.5279	0.5201	0.3907	0.4668	0.4196	0.5021	0.3310	0.4937

Table 21: Retrieval Benchmark Results on ModernBERT: MRL vs. SFS (trained with $d_{frag} = 16$)

	MRL					
Dataset	**64**	**128**	**256**	**384**	**512**	**768**
ArguAna	0.5466	0.5405	0.5565	0.5654	0.5695	0.5757
ClimateFEVER	0.1988	0.2500	0.2678	0.2578	0.2663	0.2883
DBPEDIA	0.4465	0.4457	0.4773	0.5019	0.5048	0.5108
FEVER	0.6840	0.6898	0.7248	0.7928	0.7784	0.7941
FiQA2018	0.3174	0.3484	0.4274	0.4266	0.4484	0.4603
HotpotQA	0.5407	0.5780	0.6066	0.6050	0.6402	0.6278
MSMARCO	0.5313	0.5342	0.5768	0.5613	0.5452	0.5612
NFCorpus	0.1627	0.1900	0.2050	0.2100	0.2109	0.2236
NQ	0.4743	0.4984	0.5357	0.5301	0.5238	0.5230
QuoraRetrieval	0.9286	0.9462	0.9441	0.9548	0.9479	0.9478
SCIDOCS	0.2085	0.2414	0.2544	0.2897	0.2877	0.2891
SciFact	0.4845	0.5479	0.5692	0.6103	0.6043	0.5958
Touche2020	0.4701	0.4858	0.4890	0.4955	0.5091	0.5161
	SFS					
Dataset	**64 (4)**	**128 (8)**	**256 (16)**	**384 (24)**	**512 (36)**	**768 (48)**
ArguAna	0.5015	0.5156	0.5747	0.5963	0.5953	0.5937
ClimateFEVER	0.2023	0.2347	0.3020	0.2941	0.3113	0.2994
DBPEDIA	0.3676	0.4268	0.4677	0.4916	0.4926	0.5100
FEVER	0.5891	0.7333	0.7919	0.7728	0.7990	0.8051
FiQA2018	0.3353	0.3756	0.4340	0.4606	0.4795	0.4726
HotpotQA	0.5042	0.5840	0.6196	0.6525	0.6850	0.6776
MSMARCO	0.5071	0.5264	0.5761	0.5670	0.5802	0.5755
NFCorpus	0.1226	0.1445	0.2193	0.2467	0.2338	0.2544
NQ	0.3836	0.4491	0.5341	0.5408	0.5728	0.5616
QuoraRetrieval	0.9181	0.9167	0.9357	0.9447	0.9460	0.9408
SCIDOCS	0.1633	0.1961	0.2488	0.2832	0.2956	0.2911
SciFact	0.5114	0.5646	0.5954	0.6354	0.6355	0.6640
Touche2020	0.4530	0.4729	0.4977	0.5038	0.5068	0.5201