Combining Embedding Models for RAG: A Similarity-Based Approach

Kanishka Ghosh Dastidar[1,*], Michael Dinzinger[1], Laura Caspari[1], Jelena Mitrović[1] and Michael Granitzer[1]

[1]University of Passau, Innstraße 41, 94032 Passau, Germany

Abstract

This paper explores the effective combination of embedding models through a similarity analysis. Embedding models are characterized by heterogeneous architectures, training objectives, and training corpora, resulting in distinct semantic representations and domain-specific proficiencies. By integrating such diverse models, we aim to harness their complementary strengths. However, the potential for performance improvement is contingent on the specific models selected for combination. To address this, we propose a strategy that utilizes similarity scores between model pairs as strong predictors of their combined performance. First, we provide a similarity analysis of embedding models as an unsupervised evaluation framework from a retrieval perspective. Second, we empirically demonstrate that combining embeddings from different models can significantly enhance retrieval performance, but only in the case of certain model pairs. Third, we propose an adjusted dissimilarity measure that accounts for both the similarity and performance gap between the models. Our experiments reveal that the adjusted dissimilarity scores between models are strongly correlated to their combined performance. This finding paves the way for organizations with resource constraints to use such a similarity framework to select combinations of smaller models whose performance closes the gap to large, inaccessible models.

1. Introduction

A key component of Retrieval Augmented Generation (RAG) systems is the retriever. These are typically dense retrievers, which use fixed-size numerical representations (embeddings) of the queries and documents, capturing the semantic meaning of the text. With the growing success of large language models (LLMs) in a wide range of natural language processing tasks, many institutions - both in the private and public sectors - have introduced foundational language models that can be utilized to generate embeddings for dense retrieval. Real-world RAG use-cases often involve querying corpora containing millions of documents, rendering it infeasible from a computational perspective, for certain organizations, to evaluate a large pool of models. With all but one of the top ten models for retrieval on MTEB [1] consuming over 25 Gigabytes of memory, encoding large corpora with such models would require access to powerful GPUs, which are cost-prohibitive for many organizations both within and outside

Second International Workshop on Open Web Search (WOWS 2025)

*Corresponding author.

kanishka.ghoshdastidar@uni-passau.de (K. G. Dastidar); michael.dinzinger@uni-passau.de (M. Dinzinger); laura.caspari@uni-passau.de (L. Caspari); jelena.mitrovic@uni-passau.de (J. Mitrović); michael.granitzer@uni-passau.de (M. Granitzer)

0000-0003-4171-0597 (K. G. Dastidar); 0009-0003-1747-5643 (M. Dinzinger); 0009-0002-6670-3211 (L. Caspari); 0000-0003-3220-8749 (J. Mitrović); 0000-0003-3566-5507 (M. Granitzer)

CEUR Workshop Proceedings ceur-ws.org ISSN 1613-0073

published 2025-12-18

academia [2]. A promising approach to address this arises through the ensembling of embedding models. Specifically, by combining smaller, less resource-intensive models, the performance gap to larger models could be closed.

In this paper, we focus specifically on the effective combination of embedding models through a similarity analysis. Different works in the literature have discussed the benefits of a similarity analysis of LLMs. Caspari et al. [3] argue that such an analysis of embedding models provides several benefits beyond those of the retrieval scores reported on benchmarks, such as easier model selection. Similarly, Klabunde et al. [4] highlight that similarity analysis can simplify the process of model ensembling. These works, however, do not empirically demonstrate the purported benefits.

We start from the observation that models exhibit diverse architectures, training objectives, and fine-tuning strategies, allowing them to capture distinct semantic, syntactic, and contextual aspects of text. Additionally, training on different corpora results in domain-specific capabilities. By combining these models, we aim to leverage their complementary strengths and mitigate individual biases and errors. However, the existence of improvement and its magnitude depends on the specific models involved. The embeddings might also contain conflicting semantic information – for example, architectural/training improvements in successive iterations within a model family might specifically target a particular kind of error or bias in these models. Combining such models (the predecessor and successor in a particular model family) might obscure these improvements and result in minimally improved or even impaired combined performance.

Given the possibility of beneficial or detrimental combinations and a huge number of possible combinations, a strategy to guide the selection of model combinations is essential. We posit that the similarity scores between pairs of models can serve as strong predictors of the potential performance of their combination. We do note, however, that similarity scores reflect both performance gaps along with complementarity of text representations. To predict improvements in combined performance, it is necessary to consider both of these factors that contribute to model similarity. In particular, we make the following contributions: **(i)** We provide a similarity analysis of several models from the MTEB benchmark using both representational, as well as functional similarity measures tailored for retrieval. **(ii)** We evaluate combinations of embedding models based on retrieval performance, demonstrating that performance gains are limited to specific model pairs. **(iii)** We formulate an adjusted dissimilarity measure that accounts for the performance gaps between models and demonstrate a strong correlation between this measure and combination performance. The code for this paper is publicly available[1].

2. Related Work

We review the literature from two perspectives. First, works that cover a similarity analysis of neural networks or, more specifically, language models. Second, we explore works related to ensembling or combining embeddings.

Measuring the similarity of neural networks has received increasing attention, resulting in numerous approaches or metrics being proposed [5, 6, 7, 8]. Klabunde et al. [9] provide

[1] https://github.com/KanishkaGD/embedding_similarity

an extensive overview of representational and functional similarity measures. However, the functional measures they describe are designed only for classification tasks and are thus not applicable to measure retrieval similarity. Several other works focus on evaluating the similarities of language models, primarily considering representational similarity [6, 10, 11, 12]. While a large body of research concerned with model similarity exists, embedding models are commonly compared based on their performance on benchmarks like MTEB [1]. Klabunde et al. [13] notably look beyond performance benchmarks and propose a benchmark to measure representational similarity of neural networks, including language models. Similarly, Caspari et al. [3] emphasize the importance of considering model similarity alongside benchmark retrieval scores, particularly for retrieval systems and Retrieval-Augmented Generation (RAG). They propose the use of Jaccard and rank similarity as measures to compare retrieved document lists. The work by Iana et al. [14] uses a similar methodology to analyze encoder architectures for news recommendation. They not only compare the recommendation lists in terms of performance but also similarity. Most of these works use a similarity analysis to provide qualitative insight into model behavior but do not empirically evaluate the benefits of the same.

Our work on combining the embeddings of multiple embedding models draws inspiration from the extensive work in model ensembling [15, 16, 17]. In contrast, the literature that directly explores how to best combine embeddings of different LLMs is rather sparse. Parcheta et al. [18] propose combining embeddings derived from several encoding approaches such as using BERT, a neural embedding layer, GloVe, etc., as input to a neural network. Similarly, Ghannay et al. [19] exploit the complementarity of different word embedding methods by combining embeddings, demonstrating that simple concatenation performs comparably to more complex combination methods. Some works, on the other hand, rather focus on combining embeddings of disparate inputs [20]. Xue et al. [21] leverage embeddings from two audio encoders for automatic speech recognition. Additionally, Liu et al. [22] investigate several methods to fuse embeddings from models like Llama2, RoBERTa, and BERT, highlighting the potential gains from such integrations. Tekin et al. [23] provide two different approaches to ensembling several LLMs, but this work does not focus on embeddings.

None of the works in the literature extensively evaluate combinations of several recent embedding models. To the best of our knowledge, there are also no papers that connect the two directions of this work, namely a similarity analysis and embedding combinations.

3. Similarities of Embedding Models

Our first step for finding suitable model combinations is to determine the similarity between two embedding models, as discussed in [3]. We consider two kinds of similarity metrics: (i) representational similarity, i.e. the similarity of models in embedding space and (ii) functional similarity, i.e. similarity based on observed retrieval rankings of documents from different models.

3.1. Metrics

Representational Similarity has been estimated using Centered Kernel Alignment [5]. CKA does not rely on inputs to have equal dimensions, an important facet when comparing a diverse

set of models. We use CKA to compare two sets of embeddings $E \in \mathbb{R}^{N \times D}$ and $E' \in \mathbb{R}^{N \times D'}$ generated by different embedding models for the same text chunks, where N is the number of embeddings in the set and D, D' are the respective embedding dimensions. CKA applies a kernel function to calculate scores between all entries in each set of embeddings. Thus, each row k_i of the resulting matrix for embeddings E then contains entries of the scores between embedding $e_i \in E$ and all other embeddings in E, including itself. After obtaining the kernel matrices K, K' which now have matching dimensions, CKA can be computed as:

$$CKA(K, K') = \frac{HSIC(K, K')}{\sqrt{HSIC(K, K)HSIC(K', K')}} \tag{1}$$

where HSIC is the Hilbert-Schmidt Independence Criterion [24]. The CKA score is bound in [0, 1] where 1 indicates perfect embedding similarity. In the following experiments, we use CKA with a linear kernel.

Functional Similarity has been estimated using Jaccard and rank similarity [25]. Jaccard similarity measures the overlap in retrieved results for a particular query. To this end, we perform a regular retrieval step and fetch the k most similar embeddings E, E' for a query. From this, we get two sets of retrieved text chunks C, C' which we compare with Jaccard similarity:

$$Jaccard(C, C') = \frac{|C \cap C'|}{|C \cup C'|} \tag{2}$$

The resulting score is bounded in [0, 1] with 1 indicating that both models retrieved the same text chunks. This measure disregards the order in which elements were retrieved.

Rank similarity measures how similar the order of common elements in two sets is, where elements that are closer together are weighted higher. Common text chunks $i \in |C \cap C'|$ receive a rank according to their position in the retrieval step, i.e. $r_C(i) = n$ if chunk i was the top-n retrieved result in C. With this, rank similarity is computed as follows:

$$RankSim(C, C') = \frac{1}{H(|C \cap C'|)} \sum_{j \in |C \cap C'|} \frac{2}{(1 + |r_C(j) - r_{C'}(j)|)(r_C(j) + r_{C'}(j))} \tag{3}$$

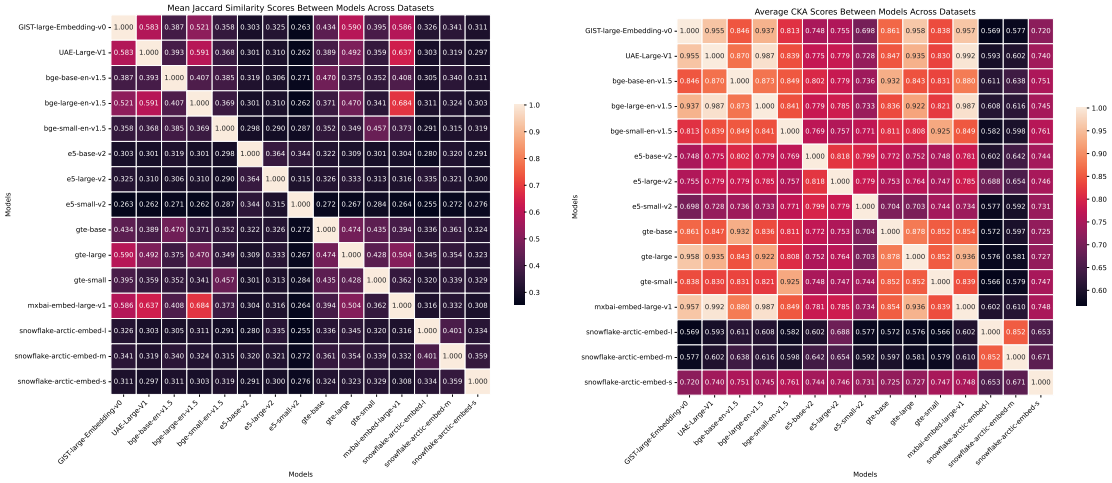
where $H(|C \cap C'|) = \sum_{k=1}^{K=|C \cap C'|} \frac{1}{k}$ denotes the K-th harmonic number and normalizes the score. Like the previous measures, rank similarity is bounded in [0, 1] with 1 indicating that the ranks of all common text chunks are identical.

3.2. Similarity Evaluation

To adhere to computational constraints, for this study, we use the 4 smallest datasets from the MTEB/BEIR [26] benchmarks, namely the Scidocs, Scifact, FIQA-2018 and Nfcorpus datasets. For each of these datasets, we evaluate pair-wise similarities of a set of 15 embedding models (publicly available and free to use). To highlight the effects of combinations within and across model families, we pick different model types from the bge (v1.5), e5 (v2), gte, and arctic model

families. We also select a few high-performing models on MTEB that are still relatively small (≈ 1 GB) such as uae-large-V1, mxbai-embed-large-V1 and gist-large-embedding-v0.

Fig. 1 displays two heatmaps, one for Jaccard and another for CKA, of the pairwise similarities of all our models averaged across our four datasets. Since the rank similarity scores closely resemble the Jaccard results, we have not included a third heatmap here. However, these scores are incorporated into our analysis in Section 5. Examining the Jaccard heatmap, the most striking feature is the relatively small overlap in retrieval results for the majority of model pairs. This indicates that *the top retrieval results for these models contain several unique documents*. We see that for each of the metrics, the similarities within families are higher than across families at least when accounting for other factors such as size. The CKA scores are fairly high across the board but are less intuitively interpretable. Notably, the similarity of the two larger arctic models with all other models is significantly lower than the average similarity according to the CKA metric. This pattern is not particularly evident with the other similarity measures.



(a) Jaccard Similarity (b) CKA

Figure 1: Mean Jaccard (left) and CKA (right) similarity scores between model pairs across all datasets (Nfcorpus, FIQA, Scidocs, Scifact).

In addition, we aim to answer the following: *How consistent are these similarity scores across datasets?* To this end, we report the correlation of these pairwise similarities across datasets in Table 1. We note that for each of our metrics, the similarity scores between model pairs are strongly associated across datasets. While a broader analysis across a diverse and larger set of data would improve the robustness of our results, they still indicate that a similarity analysis is transferable across datasets and domains.

4. Combinations of Embedding Models

In the next step, we look at the impact of combining embeddings of pairs of embedding models. There are several methodological options for combining embeddings. These lie on the spectrum of aggregation-based methods such as averaging or concatenating over multiple embeddings,

Table 1
Pearson correlation coefficients between the pairwise model similarity scores of two datasets.

	Jaccard Similarity				Rank Similarity				CKA Similarity			
	FIQA	**Nfcorpus**	**Scidocs**	**Scifact**	**FIQA**	**Nfcorpus**	**Scidocs**	**Scifact**	**FIQA**	**Nfcorpus**	**Scidocs**	**Scifact**
FIQA	1.000	0.929	0.899	0.917	1.000	0.932	0.898	0.931	1.000	0.868	0.929	0.866
Nfcorpus	0.929	1.000	0.935	0.963	0.932	1.000	0.913	0.933	0.868	1.000	0.953	0.994
Scidocs	0.899	0.935	1.000	0.958	0.898	0.913	1.000	0.938	0.929	0.953	1.000	0.963
Scifact	0.917	0.963	0.958	1.000	0.931	0.933	0.938	1.000	0.866	0.994	0.963	1.000

or methods that involve some form of learned combination where the combined embedding is optimized on task-specific objectives.

4.1. Our Approach

We take the simple approach of concatenating the embeddings of pairs of models for both queries and documents. We then evaluate these combined embeddings on different retrieval tasks from the MTEB benchmark. We use the same set of embedding models and datasets as outlined in 3.

For a dataset comprising a set of queries $\mathcal{Q} = \{q_j\}_{j=1}^{M}$ and a set of chunks $\mathcal{C} = \{c_k\}_{k=1}^{N}$, the embeddings from two models $\mathcal{E}_1$ and $\mathcal{E}_2$ are concatenated to form a unified or combined representation:

$$\mathbf{E}_{\text{query}}(q_j) = \mathcal{E}_1(q_j) \| \mathcal{E}_2(q_j) \quad \forall\, q_j \in \mathcal{Q}$$
$$\mathbf{E}_{\text{chunk}}(c_k) = \mathcal{E}_1(c_k) \| \mathcal{E}_2(c_k) \quad \forall\, c_k \in \mathcal{C}$$

For the retrieval step, we use an exact nearest neighbour search. We compute the Normalized Discounted Cumulative Gain at 10 documents (NDCG_{10}) to evaluate retrieval performance. As each chunk in our retrieval list stems from a document in our corpus, we can compute the performance for a list of documents instead of a list of chunks as shown by [27]. This is necessary as the relevance judgments for these datasets are at a document and not chunk level.

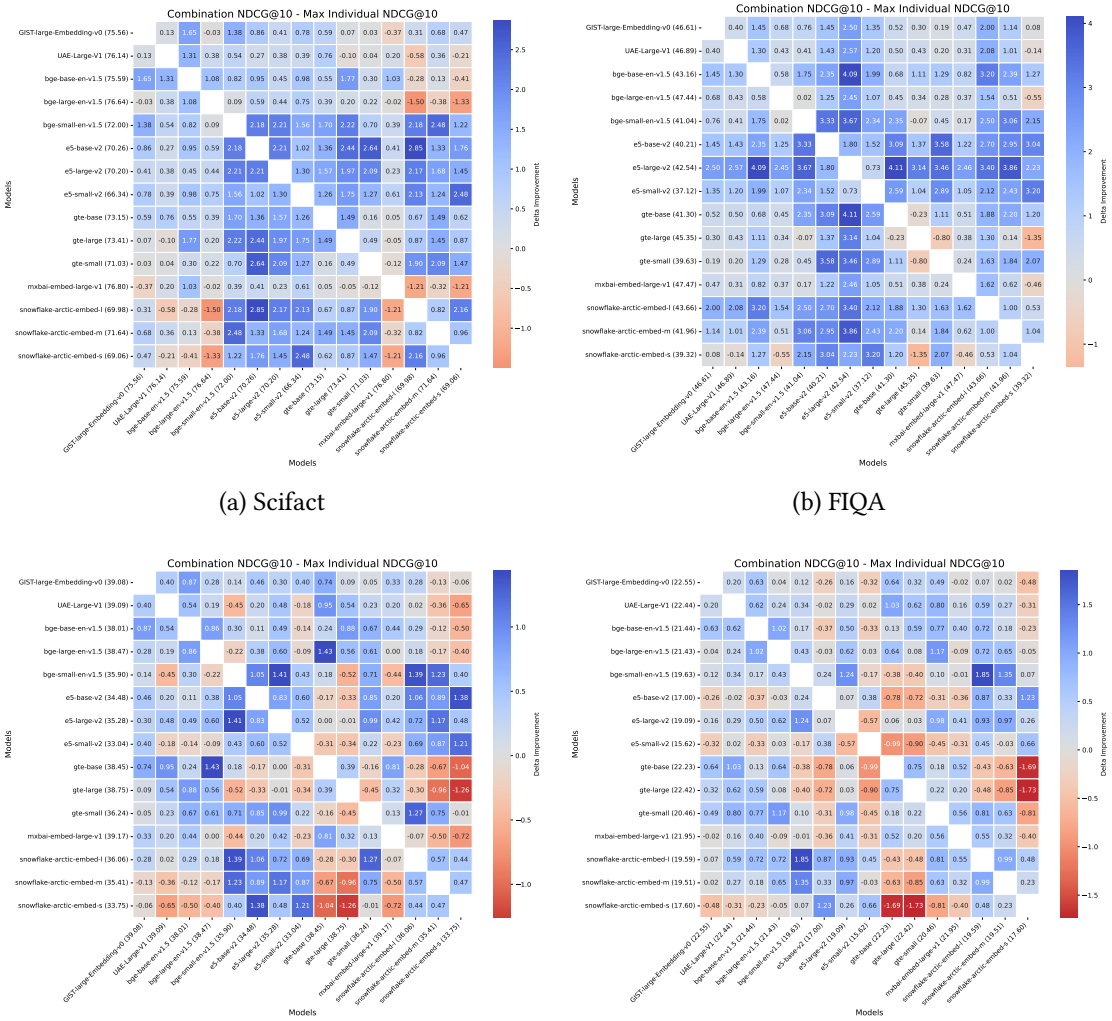
Of course, while the NDCG_{10} directly informs us about how well a particular combination performs, the score is only of use relative to the scores of the two individual models that form the combination. From a practical perspective, there is little justification for utilizing a combination of models if its performance does not surpass that of both individual models. Consequently, in our analysis, we adopt a metric for measuring performance improvement ($\mathcal{I}_{\text{ndcg}}$). Specifically, we define performance improvement as the difference between the NDCG_{10} score of the combined embedding model and the maximum NDCG_{10} score of the individual models within the pair.

$$\mathcal{I}_{\text{ndcg}} = \text{NDCG}_{10}(\mathcal{E}_1 \| \mathcal{E}_2) - \max\left(\text{NDCG}_{10}(\mathcal{E}_1), \text{NDCG}_{10}(\mathcal{E}_2)\right)$$

In the remainder of this paper, we refer to the maximum of the two individual models' scores as Max_{comb}. We note that the scores are not perfectly comparable to those on the MTEB leaderboard due to differences in our pipeline. The first stems from our data ingestion step, where we split the documents into 256 token-sized chunks for each of our models. This results in relevant information contained in the documents possibly being fragmented across several chunks. Secondly, as we test on small corpora and want to include as many queries as possible, we report retrieval scores on all queries in the datasets and not just those included in the test

splits. This being said, the scores we report largely preserve the ranking of the models as reported on MTEB.

4.2. Results



(a) Scifact (b) FIQA

(c) Nfcorpus (d) Scidocs

Figure 2: Improvement in performance through model combination. The difference in NDCG_{10} refers to comparing the performance of two sets of embeddings for the retriever: one where the embeddings of two models are combined and another where only the embeddings of better-performing model of the two is used.

In Figure 2 we report the $\mathcal{I}_{\text{ndcg}}$ values for all combinations of our selected models on four datasets. Across each dataset, $\mathcal{I}_{\text{ndcg}}$ is predominantly positive, indicating that combining a randomly selected pair of models generally leads to improved retrieval performance compared to Max_{comb}. This re-affirms our argument that these models, in many instances, generate

heterogeneous representations that complement one another. However, it is also apparent that the magnitude of improvement or even impairment, is dependent on a particular combination and dataset. To begin, we assess the influence of different model pair selections on the $\mathcal{I}_{\text{ndcg}}$ scores. For example, the e5-large-v2 model tends to result in positive $\mathcal{I}_{\text{ndcg}}$ values for nearly all of its combinations across all four datasets. Similarly, combinations involving the model bge-small-en-v1.5 mostly result in positive $\mathcal{I}_{\text{ndcg}}$. Such performance gains can provide significant benefits. For example, for the FIQA dataset the combination of the e5-small-v2 and snowflake-arctic-embed-s models, both of whom are very small models (0.12 GB in memory usage each), result in a NDCG_{10} higher than that of the e5-large-v2 model. This would entail a climb on the public leaderboard corresponding to around 10 positions. We also observe large positive $\mathcal{I}_{\text{ndcg}}$ values in the combinations of bge-small-en-v1.5 and snowflake-arctic-embed-m across all our datasets.

Certain combinations, on the other hand, do not fare as well. The Arctic model family, particularly the small version, is also often present in combinations with a negative $\mathcal{I}_{\text{ndcg}}$. These combinations are of no practical benefit as one could simply use the better performing model in the combination. To a lesser extent, for the Nfcorpus and Scidocs datasets, we see a similar effect for the e5-small model. We note that these models are also the ones with the lowest retrieval performance out of our set of models. Assuming that the 'general' quality of the representations generated by these models corresponds to their retrieval scores, we can reasonably expect significant deficiencies of certain lower performing models to have a negative impact on the overall retrieval performance when combined with stronger counterparts. These findings indicate that both the complementarity of the model representations and the disparities in their quality significantly influence the effectiveness of their combined retrieval performance. Across the analyzed datasets, we observe that certain datasets not only exhibit a higher proportion of model combinations yielding positive $\mathcal{I}_{\text{ndcg}}$ values but also demonstrate larger magnitudes of these improvements. For the Scidocs dataset, 71 combinations out of 105 improve over their Max_{comb}, whereas for FIQA the number is 98. The magnitude of improvements is also on average higher on FIQA and Scifact compared to the other two datasets.

5. Similarity-Guided Combination

We hypothesize that enhancements in retrieval performance resulting from model combinations are attributable to their complementary differences. Similarity scores provide a quantitative means to compare model characteristics — the differences in architecture, training, data, etc. should reflect in the orientations of the embeddings in vector space (captured by representational similarity measures) and/or in the outcome on downstream tasks (captured by functional similarity measures). Therefore, we ask the question: *Can we use these similarity scores as predictors of the compatibility of models to be combined*? Assuming that models need to be sufficiently distinct to be complementary, similarity metrics that accurately capture these differences would allow us to identify such distinct model pairs. The following parts describe our framework to test this hypothesis.

5.1. Our Approach

We use our similarity scores and $\mathcal{I}_{\text{ndcg}}$ values from sections 3 and 4. Our central hypothesis posits that the performance gains from combining two models are positively correlated with their dissimilarity; i.e., more dissimilar models are expected to yield greater improvements when combined. Therefore, we transform our existing similarity metrics into dissimilarity scores. Given that each of our three similarity metrics is normalized to lie within the interval $[0, 1]$, the transformation is straightforward:

$$D_{ij} = 1 - S_{ij} \tag{4}$$

where D_{ij} and S_{ij}denotes the dissimilarity and similarity between models $\mathcal{E}_i$ and $\mathcal{E}_j$ respectively.

However, a critical observation arises: the raw dissimilarity scores are intrinsically coupled with the performance gap between model pairs. Specifically, a model exhibiting superior performance ($\mathcal{P}_i$) is inherently more dissimilar to models with significantly lower performance ($\mathcal{P}_j$) compared to models with similar performance levels. Our similarity analysis in Section 3 confirms this. This coupling poses a confounding challenge, as the observed dissimilarity may partially reflect performance disparities rather than purely complementary representations of the text. To predict improvements in combined performance, we would need to account for both these factors that contribute to dissimilarity.

To address this, we introduce an **Adjusted Dissimilarity** metric. This metric modifies the dissimilarity score between models by accounting for the performance gap between them. We define the normalized performance gap between models $\mathcal{E}_i$ and $\mathcal{E}_j$ as follows:

$$\overline{\mathcal{PG}}_{ij} = \frac{|\mathcal{P}_i - \mathcal{P}_j|}{\mathcal{PG}_{\text{max}}} \tag{5}$$

where $\mathcal{PG}_{\text{max}}$ is defined as the maximum absolute performance difference observed across all possible pairs of models:
The **Adjusted Dissimilarity** AD_{ij} is then defined as:

$$\text{AD}_{ij} = \alpha \times D_{ij} + (1 - \alpha) \times (1 - \overline{\mathcal{PG}}_{ij}) \tag{6}$$

where α is a parameter that balances the influence of the raw dissimilarity and the normalized performance gap on the adjusted dissimilarity score. Intuitively, the term $(1 - \overline{\mathcal{PG}}_{ij})$ inversely relates to the performance gap. When the performance gap is large, $(1 - \overline{\mathcal{PG}}_{ij})$ becomes smaller, reducing the contribution of this term to overall adjusted dissimilarity.

Correlation Analysis: For each model $\mathcal{E}_i$, we compute the adjusted dissimilarity scores AD_{ij} with every other model $\mathcal{E}_j$, as defined in Equation (6). Similarly, we compute the $\mathcal{I}_{\text{ndcg}}$ values, resulting from combining $\mathcal{E}_i$ with $\mathcal{E}_j$. Given these values, for each model $\mathcal{E}_i$, we compute Pearson's correlation coefficient ρ^k between the set of its Adjusted Dissimilarity scores $\{\text{AD}_{ij}\}$ and the corresponding $\mathcal{I}_{\text{ndcg}}$ values $\{\mathcal{I}_{\text{ndcg}}\}$ across all model pairs $(\mathcal{E}_i, \mathcal{E}_j)$ for a particular similarity metric k. This measures the linear relationship between dissimilarity and performance improvement. After computing ρ^k for each model, we aggregate these coefficients by calculating the mean correlation coefficient $\overline{\rho}^{(k)}$ across all models. We perform this analysis per model to offer more granular insight into model-specific behaviour. We also report an overall correlation coefficient $\rho^{(k)}_{\text{overall}}$ per metric for all model pairs for a dataset.

5.2. Results

For each of our datasets, we provide a detailed overview of our correlation analysis in Table 2. As per Eq. 6, we need to set a value for the α parameter to compute the adjusted dissimilarity scores. For this study we use α values of 0.7 for Jaccard, 0.5 for CKA, and 0.8 for rank similarity. These values were tuned on Nfcorpus and then kept constant for each of the other datasets. This would indicate, contingent on the results across datasets, that these alpha values could serve as general recommendations for each metric.

Table 2
Correlation analysis: Each cell in the table displays the Pearson correlation coefficient of a particular model's $\mathcal{I}_{\text{ndcg}}$ values and adjusted dissimilarity scores for a particular metric. Also displayed is the overall coefficient for a particular metric k ($\rho^{(k)}_{\text{overall}}$), computed for all model pairs on a dataset.

Model	**Nfcorpus**				**SciFact**			
	Baseline	Jaccard	CKA	Rank Sim.	Baseline	Jaccard	CKA	Rank Sim.
bge-large-en-v1.5	0.579	**0.739**	0.563	0.720	0.233	0.117	0.004	**0.436**
bge-base-en-v1.5	0.789	0.844	0.650	**0.918**	0.565	0.519	0.375	**0.673**
bge-small-en-v1.5	0.886	**0.950**	0.876	0.927	0.640	0.910	0.783	**0.922**
GIST-large-Embedding-v0	0.340	0.639	0.269	**0.676**	-0.112	0.739	0.070	0.727
UAE-Large-V1	0.654	**0.689**	0.567	0.601	0.353	0.442	0.260	**0.622**
e5-base-v2	0.836	0.911	0.841	**0.927**	0.871	0.883	0.852	**0.883**
e5-small-v2	0.864	0.902	0.900	**0.904**	0.772	**0.854**	0.796	0.846
e5-large-v2	0.801	0.893	0.767	**0.902**	**0.927**	0.899	0.903	0.915
arctic-embed-l	0.766	**0.904**	0.762	0.881	0.907	**0.968**	0.933	0.965
arctic-embed-m	0.876	0.844	0.861	**0.930**	0.831	**0.945**	0.868	0.938
arctic-embed-s	0.898	**0.923**	0.876	0.890	0.907	0.937	0.897	**0.938**
mxbai-embed-large-v1	0.619	**0.622**	0.528	0.499	0.196	0.156	-0.006	**0.381**
gte-large	**0.829**	0.793	0.742	0.699	0.017	0.804	0.356	**0.836**
gte-base	0.816	0.898	0.665	**0.916**	0.251	0.665	0.408	**0.727**
gte-small	0.774	**0.954**	0.794	0.950	0.759	0.967	0.846	**0.973**
Mean ($\overline{\rho}^{(k)}$)	0.752	**0.841**	0.703	0.823	0.541	0.720	0.557	**0.785**
Overall ($\rho^{(k)}_{\text{overall}}$)	0.671	**0.819**	0.646	0.801	0.500	0.779	0.570	**0.838**
Model	**Scidocs**				**FIQA**			
	Baseline	Jaccard	CKA	Rank Sim.	Baseline	Jaccard	CKA	Rank Sim.
bge-large-en-v1.5	0.453	**0.867**	0.683	0.811	0.011	**0.632**	0.154	0.624
bge-base-en-v1.5	0.829	**0.916**	0.834	0.886	0.577	0.936	0.794	**0.946**
bge-small-en-v1.5	0.813	0.899	**0.903**	0.881	0.756	0.967	0.871	**0.968**
GIST-large-Embedding-v0	0.736	**0.766**	0.707	0.583	-0.093	0.852	0.144	**0.866**
UAE-Large-V1	0.612	**0.827**	0.687	0.778	-0.074	**0.808**	0.143	0.799
e5-base-v2	0.822	**0.870**	0.870	0.803	0.887	0.946	0.920	**0.951**
e5-small-v2	0.646	**0.721**	0.686	0.696	0.755	0.857	0.819	**0.881**
e5-large-v2	0.821	0.823	0.812	**0.849**	0.793	**0.882**	0.840	0.846
arctic-embed-l	0.684	**0.887**	0.646	0.829	0.363	0.846	0.504	**0.895**
arctic-embed-m	0.799	**0.877**	0.724	0.786	0.672	0.940	0.761	**0.943**
arctic-embed-s	0.789	**0.820**	0.786	0.745	0.849	**0.950**	0.875	0.948
mxbai-embed-large-v1	0.598	0.751	**0.794**	0.625	-0.037	**0.692**	0.123	0.689
gte-large	**0.829**	0.649	0.676	0.532	0.257	**0.735**	0.428	0.701
gte-base	**0.876**	0.829	0.751	0.805	0.642	0.930	0.768	**0.933**
gte-small	**0.744**	0.636	0.609	0.483	0.681	0.941	0.791	**0.949**
Mean ($\overline{\rho}^{(k)}$)	0.737	**0.809**	0.744	0.739	0.469	0.861	0.596	**0.863**
Overall ($\rho^{(k)}_{\text{overall}}$)	0.692	**0.746**	0.708	0.673	0.452	0.881	0.635	**0.884**

Does using the similarity scores improve over simply using the performance gap as a measure? As the objective is to compare the metrics, we focus on the values across columns for each dataset. For the first column, we define the baseline as the correlation of how close the models

are in terms of performance (by setting $\alpha = 0$ in Eq. 6) to the $\mathcal{I}_{\text{ndcg}}$ values. This would be the case of simply picking the two models that are closest in terms of retrieval performance to combine. We argue that this is a fair baseline given that we measure the improvement in retrieval performance over the better model. Using this strategy, we see a moderate positive association between our baseline scores and $\mathcal{I}_{\text{ndcg}}$ values, with the highest correlation reported for the Nfcorpus dataset. For the adjusted dissimilarity scores to be of practical value, the correlation scores for a particular metric would have to be higher than the baseline. We see that this is not the case when using CKA as a metric, with nearly identical $\overline{\rho}^{(k)}$ and $\rho^{(k)}_{\text{overall}}$ values as the baseline for all our datasets (with the exception of FIQA where it is better). When it comes to the two functional similarity measures, Jaccard and rank similarity, we see an improvement over the baseline in terms of a stronger positive association (both $\overline{\rho}^{(k)}$ and $\rho^{(k)}_{\text{overall}}$) on almost all datasets. The only exception being Scidocs for rank similarity. For Scifact and FIQA, both these metrics show a significantly higher association than the baseline.

In addition to the results presented in the table, we also computed $\rho^{(k)}_{\text{overall}}$ for each of FIQA, Scifact and Scidocs while using the similarity scores computed on NFcorpus. The resulting coefficients were 0.784, 0.758 and 0.759 respectively. This re-affirms our findings in Section 3 that the similarity scores are indeed consistent and need not be necessarily re-computed for each dataset.

In terms of the more granular model-specific view, the majority of models show a strong association between their adjusted dissimilarity scores (excluding CKA) and the $\mathcal{I}_{\text{ndcg}}$ values. However, there is a trend of lower correlation values the larger the model is. Why this is the case must be further investigated and can be considered as a current limitation of our analysis.

6. Conclusion

In this paper, we present several key findings. **First**, our similarity analysis reveals significant differences in both the embedding spaces learned by various models and the retrieved lists they generate. Importantly, these similarities between model pairs remain consistent across different datasets, allowing practitioners to avoid repeating similarity evaluations for every new task. **Second**, our extensive evaluation of over a hundred model pairs demonstrates that, in certain cases, concatenating embeddings leads to substantial improvements in retrieval performance. These enhancements can propel models several positions higher on the MTEB leaderboard. However, we also identify model pairs that show no improvement or even a decline in performance, underscoring the necessity for a strategic approach in selecting model combinations. **Third**, to address this, we leverage our similarity analysis by transforming similarity scores to an adjusted dissimilarity measure that accounts for the performance gap between models. Our results indicate a strong correlation between performance improvements and this measure, particularly when using Jaccard and rank similarity metrics. Consequently, practitioners can utilize model similarities to inform their decisions on combining embeddings, thereby reducing the need to evaluate every possible combination manually.

Looking towards the future, we aim to expand our similarity evaluation and provide a public benchmark for the same. Additionally, we plan to use our evaluation framework to deploy promising combinations of smaller models publicly.

7. Acknowledgments

This work is part of OpenWebSearch.eu, funded by the EU under GA 101070014, and part of CAROLL, funded by the German Federal Ministry of Education and Research (BMBF) under 01|S20049.

Declaration on Generative AI

During the preparation of this work, the author(s) used ChatGPT in order to: Improve writing style, Paraphrase and reword. After using this tool/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the publication's content.

References

[1] N. Muennighoff, N. Tazi, L. Magne, N. Reimers, Mteb: Massive text embedding benchmark, arXiv preprint arXiv:2210.07316 (2022). URL: https://arxiv.org/abs/2210.07316. doi:10.48550/ARXIV.2210.07316.

[2] S. Luccioni, Y. Jernite, E. Strubell, Power hungry processing: Watts driving the cost of ai deployment?, in: Proceedings of the 2024 ACM Conference on Fairness, Accountability, and Transparency, FAccT '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 85–99. URL: https://doi.org/10.1145/3630106.3658542. doi:10.1145/3630106.3658542.

[3] L. Caspari, K. G. Dastidar, S. Zerhoudi, J. Mitrovic, M. Granitzer, Beyond benchmarks: Evaluating embedding model similarity for retrieval augmented generation systems, 2024. URL: https://arxiv.org/abs/2407.08275. arXiv:2407.08275.

[4] M. Klabunde, T. Schumacher, M. Strohmaier, F. Lemmerich, Similarity of neural network models: A survey of functional and representational measures, arXiv preprint arXiv:2305.06329 (2023).

[5] S. Kornblith, M. Norouzi, H. Lee, G. Hinton, Similarity of neural network representations revisited, in: K. Chaudhuri, R. Salakhutdinov (Eds.), Proceedings of the 36th International Conference on Machine Learning, volume 97 of *Proceedings of Machine Learning Research*, PMLR, 2019, pp. 3519–3529. URL: https://proceedings.mlr.press/v97/kornblith19a.html.

[6] A. S. Morcos, M. Raghu, S. Bengio, Insights on representational similarity in neural networks with canonical correlation, 2018. URL: https://arxiv.org/abs/1806.05759. arXiv:1806.05759.

[7] Y. Li, J. Yosinski, J. Clune, H. Lipson, J. Hopcroft, Convergent learning: Do different neural networks learn the same representations?, 2016. URL: https://arxiv.org/abs/1511.07543. arXiv:1511.07543.

[8] Y. Bansal, P. Nakkiran, B. Barak, Revisiting model stitching to compare neural representations, Advances in neural information processing systems 34 (2021) 225–236.

[9] M. Klabunde, T. Schumacher, M. Strohmaier, F. Lemmerich, Similarity of neural network models: A survey of functional and representational measures, arXiv preprint arXiv:2305.06329 (2023).

[10] J. M. Wu, Y. Belinkov, H. Sajjad, N. Durrani, F. Dalvi, J. Glass, Similarity analysis of contextual word representation models, 2020. URL: https://arxiv.org/abs/2005.01172. arXiv:2005.01172.
[11] M. Freestone, S. K. K. Santu, Word embeddings revisited: Do llms offer something new?, 2024. URL: https://arxiv.org/abs/2402.11094. arXiv:2402.11094.
[12] D. Brown, C. Godfrey, N. Konz, J. Tu, H. Kvinge, Understanding the inner workings of language models through representation dissimilarity, 2023. URL: https://arxiv.org/abs/2310.14993. arXiv:2310.14993.
[13] M. Klabunde, T. Wald, T. Schumacher, K. Maier-Hein, M. Strohmaier, F. Lemmerich, Resi: A comprehensive benchmark for representational similarity measures, 2024. URL: https://arxiv.org/abs/2408.00531. arXiv:2408.00531.
[14] A. Iana, G. Glavaš, H. Paulheim, Peeling back the layers: An in-depth evaluation of encoder architectures in neural news recommenders, 2024. URL: https://arxiv.org/abs/2410.01470. arXiv:2410.01470.
[15] L. Breiman, Bagging predictors, Machine learning 24 (1996) 123–140.
[16] L. Breiman, Random forests, Machine learning 45 (2001) 5–32.
[17] K. M. Ting, I. H. Witten, Stacking bagged and dagged models (1997).
[18] Z. Parcheta, G. Sanchis-Trilles, F. Casacuberta, R. Rendahl, Combining embeddings of input data for text classification, Neural Processing Letters 53 (2021) 3123–3151. URL: https://doi.org/10.1007/s11063-020-10312-w. doi:10.1007/s11063-020-10312-w.
[19] S. Ghannay, B. Favre, Y. Estève, N. Camelin, Word embedding evaluation and combination, in: N. Calzolari, K. Choukri, T. Declerck, S. Goggi, M. Grobelnik, B. Maegaard, J. Mariani, H. Mazo, A. Moreno, J. Odijk, S. Piperidis (Eds.), Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16), European Language Resources Association (ELRA), Portorož, Slovenia, 2016, pp. 300–305. URL: https://aclanthology.org/L16-1046.
[20] B. Koloski, S. Pollak, R. Navigli, B. Škrlj, Automl-guided fusion of entity and llm-based representations for document classification, 2024. URL: https://arxiv.org/abs/2408.09794. arXiv:2408.09794.
[21] H. Xue, W. Ren, X. Geng, K. Wei, L. Li, Q. Shao, L. Yang, K. Diao, L. Xie, Ideal-llm: Integrating dual encoders and language-adapted llm for multilingual speech-to-text, 2024. URL: https://arxiv.org/abs/2409.11214. arXiv:2409.11214.
[22] C. Liu, H. Zhang, K. Zhao, X. Ju, L. Yang, Llmembed: Rethinking lightweight llm's genuine function in text classification, 2024. URL: https://arxiv.org/abs/2406.03725. arXiv:2406.03725.
[23] S. F. Tekin, F. Ilhan, T. Huang, S. Hu, L. Liu, Llm-topla: Efficient llm ensemble by maximising diversity, 2024. URL: https://arxiv.org/abs/2410.03953. arXiv:2410.03953.
[24] A. Gretton, O. Bousquet, A. Smola, B. Schölkopf, Measuring statistical dependence with hilbert-schmidt norms, in: S. Jain, H. U. Simon, E. Tomita (Eds.), Algorithmic Learning Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, 2005, pp. 63–77.
[25] C. Wang, W. Rao, W. Guo, P. Wang, J. Liu, X. Guan, Towards understanding the instability of network embedding, IEEE Transactions on Knowledge and Data Engineering 34 (2022) 927–941. doi:10.1109/TKDE.2020.2989512.
[26] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, I. Gurevych, BEIR: A heterogeneous

benchmark for zero-shot evaluation of information retrieval models, in: Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021. URL: https://openreview.net/forum?id=wCu6T5xFjeJ.

[27] M. Günther, I. Mohr, B. Wang, H. Xiao, Late chunking: Contextual chunk embeddings using long-context embedding models, 2024. URL: https://arxiv.org/abs/2409.04701. arXiv:2409.04701.