Embedding-Converter: A Unified Framework for Cross-Model Embedding Transformation

Jinsung Yoon, Sercan Ö. Arık
Google Cloud AI, CA, USA
{jinsungyoon, soarik}@google.com

Abstract

Embedding models play a crucial role in machine learning. However, the continuous development of new models presents a major challenge: migrating to a potentially superior model often requires the computationally expensive process of re-embedding entire datasets—without any guarantee of performance improvement. This paper presents Embedding-Converter, a novel framework for efficiently transforming embeddings between different models, thus avoiding costly 're-embedding'. The proposed approach achieves 100 times faster and cheaper computations in real-world applications. Experiments show that Embedding-Converter not only streamlines transitions to new models, but can also improve upon the source model's performance, approaching that of the target model. This facilitates efficient evaluation and broader adoption of new embedding models by significantly reducing the overhead of model switching. Furthermore, Embedding-Converter addresses latency limitations by enabling the use of smaller models for online tasks while still benefiting from the performance of larger models offline. By promoting the release of converters alongside new embedding models, Embedding-Converter fosters a more dynamic and accessible ecosystem for embedding model development and deployment.

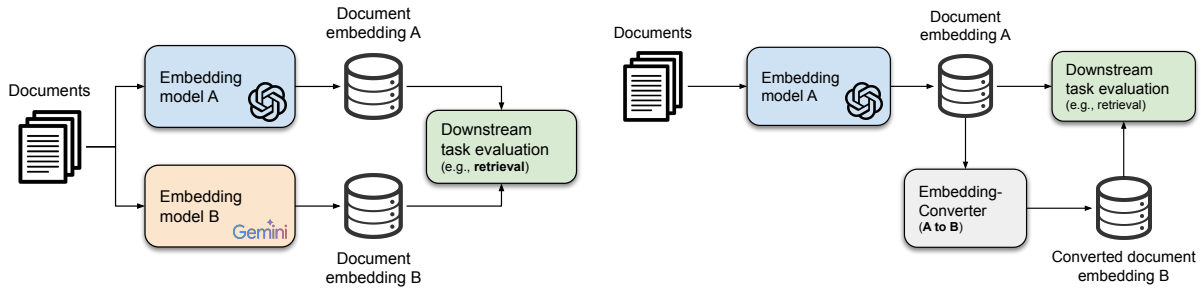
1 Introduction

Embeddings, numerical vector representations of complex data like text and images, are fundamental to modern machine learning. These representations empower a wide range of applications, including search, clustering to anomaly detection, classification, and information retrieval (Wang et al., 2016; Huang et al., 2020; Zhai et al., 2019). However, the rapidly evolving landscape of embedding models, each with unique strengths and weaknesses (Wang et al., 2022; Li et al., 2023; Lee et al., 2024a), presents a significant challenge. Choosing the best model for a given task often requires computationally intensive and time-consuming evaluation, especially with massive datasets. For instance, selecting the best model for a billion text passages necessitates generating embeddings for all passages with each candidate, a computationally-daunting task (see Appendix A for details). This challenge is exacerbated by the continuous introduction of new, potentially superior models, creating the need for repeated re-embedding without even guaranteed improvements with the new models. Moreover, the incompatibility between different embedding models, even within the same family (e.g., Google's Gecko (Lee et al., 2024b) or OpenAI's embeddings (Neelakantan et al., 2022)), requires complete re-embedding when exploring new models or upgrading existing ones. This laborious process significantly impedes efficient experimentation and the adoption of state-of-the-art embedding techniques in real-world applications.

This paper introduces Embedding-Converter, a canonical framework designed to overcome the above challenges by enabling seamless transitions between embedding models. Acting as a universal translator for embedding spaces, Embedding-Converter empowers embedding users to effortlessly explore new models, upgrade to newer versions, and even switch between entirely different model families without the computational burden of re-embedding data (see Fig. 1).

Building such a converter presents significant challenges, including learning an efficient mapping between potentially disparate high-dimensional spaces using unlabeled text data (see Fig. 2(b)). The model must balance sufficient capacity for effective transfer with the avoidance of overfitting, and training requires carefully chosen loss functions to ensure accurate conversion. This paper details the novel methodological approaches used to develop Embedding-Converter and demonstrates

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 25464–25482



(a) Conventional evaluation framework

(b) Proposed evaluation framework (Embedding-Converter)

Figure 1: Consider evaluating two embedding models, A and B, which could be different versions of the same model family or entirely distinct models. (a) The conventional approach requires re-embedding the entire corpus to evaluate model B, incurring substantial computational cost. (b) Embedding-Converter efficiently transforms existing embeddings from model A to model B, drastically reducing this overhead.

its efficacy through extensive experiments across diverse scenarios. Our evaluation includes intra-model (between versions of the same model), inter-model, and cross-dimensional conversions. Furthermore, we assess Embedding-Converter's performance on various downstream embedding tasks such as retrieval and semantic textual similarity.

The key contributions are summarized below:

- We introduce Embedding-Converter, a unified framework for cost-effective conversion between embedding models.
- It drastically reduces the computational overhead of migrating between models, enabling faster exploration and adoption of new techniques (over 100x reductions in computation cost and time).
- Our experiments consistently show Embedding-Converter's converted embeddings *closely approximate* the target and surpass source model performance on various downstream tasks, maintaining near-target model accuracy.

2 Related Work

2.1 Embedding models

Embedding models are essential for numerous applications, including information retrieval, search, recommendations, clustering and data visualization. The field is rapidly advancing, with continually-improved models from OpenAI (Neelakantan et al., 2022), Google (Lee et al., 2024b) and many other companies. This is reflected in the competitive MTEB leaderboard (Muennighoff et al., 2022), where models like NV-Embed (Lee et al., 2024a), SFR-Embedding (Rui Meng, 2024), and GTE-Qwen (Li et al., 2023) frequently update to achieve top rankings. Academia also contributes actively, with models like General Text Embedding (GTE) (Li et al., 2023) and Generalizable T5-based dense Retrievers (GTR) (Ni et al., 2021), alongside multi-modal embeddings like CLIP (Radford et al., 2021) and CoCA (Yu et al., 2022). This rapid progress and model diversity, however, create incompatibility issues, even between versions of the same model. As the MTEB leaderboard demonstrates, selecting the best model for a given task or dataset often requires evaluating multiple models, a computationally expensive and time-consuming process, especially with large datasets, due to the need for repeated re-embedding. This paper addresses this challenge by introducing Embedding-Converter, a canonical framework for seamless transitions between embedding models without requiring complete recomputation. Embedding-Converter is designed to empower practitioners to easily evaluate and migrate between models, promoting greater flexibility and efficiency in developing and deploying embedding-based applications.

2.2 Vector space transformation

Converting embeddings between models can be viewed as a vector space transformation problem, aiming to map vectors from one space to another. While linear algebra offers various solutions, including linear transformations (Marcus, 1971), change of basis (Shores et al., 2007), and kernel methods (Treves, 2013), these often assume undefined target spaces, unlike the pre-trained embedding models we consider.

Cross-lingual embedding mapping, as explored by Artetxe et al. (2017) and Conneau et al. (2017), focuses on aligning word embedding spaces across languages. While related, these methods primar-

ily address word-level embeddings, limiting their direct applicability to sentence or passage embeddings, that can capture richer semantic context. Domain adaptation research, including works like Wang et al. (2021), Schopf et al. (2023), and Yoon et al. (2024), adapts embeddings between domains. However, these approaches are often tailored to specific scenarios. In contrast, Embedding-Converter offers a more versatile, general-purpose solution for converting sentence embeddings between any model, irrespective of domain or task, setting it apart from previous work.

Although, the relevant computer vision research explores model compatibility, the proposed methods differ significantly. Backward Compatible Training (BCT) (Shen et al., 2020; Hu et al., 2022) and Forward Compatible Training (FCT) (Ramanujan et al., 2022) require modifying the training process of new models or rely on unavailable "side information", respectively, which are infeasible with fixed, pre-trained models. While Jaeckle et al. (2023) addresses some limitations, it mainly focuses on online backfilling with different data requirements and objectives. Furthermore, these methods primarily target images, while Embedding-Converter demonstrates broader applicability across various data modalities.

3 Methods: Embedding-Converter

This section details Embedding-Converter, our framework for efficiently converting embeddings between models. While demonstrated here for text embeddings, the framework is versatile and applicable to various data types, including images and multimodal data. Critically, Embedding-Converter operates with any embedding model, even those accessible only through prediction APIs with undisclosed internal workings. This broadens its applicability significantly, as many embedding models are exclusively available via prediction-only API access.

3.1 Problem formulation

Our goal is to learn a transformation between two high-dimensional embedding spaces. Specifically, we aim to convert text embeddings generated by a source model, f, to be consistent with a target model, g. Given a text passage $t \in \mathcal{T}$ (where $\mathcal{T}$ is the set of all text passages), we seek a converter function h such that $h(f(t)) \simeq g(t)$. This function maps embeddings from the source space $\mathbb{R}^{d_f}$ to the target space $\mathbb{R}^{d_g}$, where d_f and d_g are the respective embedding dimensions.

Using a corpus of unlabeled text data, $\mathcal{D} = \{t_1, t_2, ..., t_N\}$, we learn the optimal converter h. Crucially, this method doesn't require labeled data depicting inter-passage relationships. Any text corpus (e.g., MSMarco (Bajaj et al., 2016) or Wikipedia) can be used. The objective is to find the h that maximizes the similarity (e.g., cosine similarity) between the converted and target embeddings for any text t.

The proposed converter h is a unified model designed to handle any text $t \in \mathcal{T}$, regardless of dimensionality differences between the source and target spaces. However, a distinct converter h is learned for each source-target model pair. This approach enables flexible use of various embedding models by facilitating seamless transitions between their respective spaces.

3.2 Loss functions

A simple approach to maximize similarity between converted and target embeddings is to use a regression loss, minimizing the distance between embedding vectors:

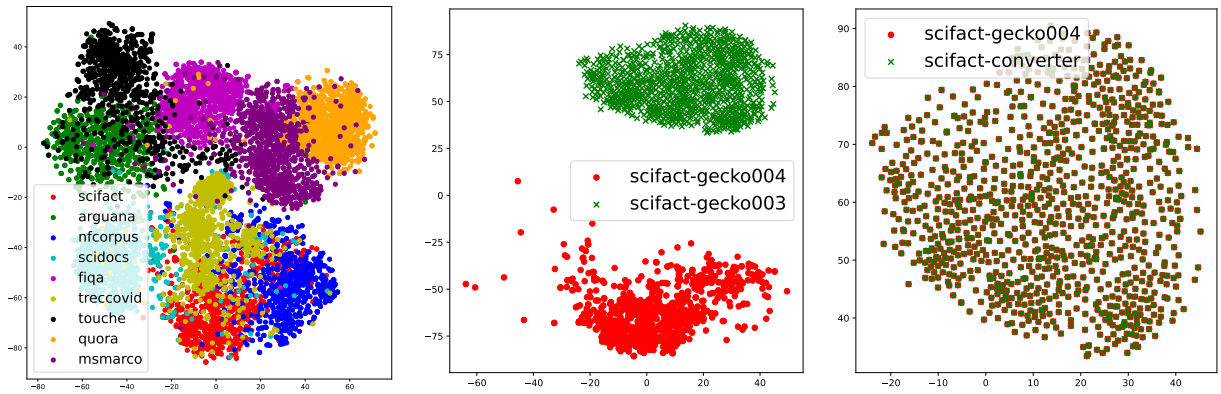
$$\mathcal{L}_{reg} = \sum_{t=1}^{N} ||h(f(t)) - g(t)||_1. \quad (1)$$

While we use mean absolute error here, other regression losses (e.g., mean squared error) could also be used. However, as our ablation study (Table 6) shows, regression loss alone is insufficient. Therefore, we introduce two additional loss functions to preserve both global and local relationships within the embedding spaces, improving conversion fidelity.

The first, a global similarity loss (similar to Park et al. (2019)), maintains overall embedding distances:

$$\mathcal{L}_{global} = \sum_{t_1,t_2 \in \mathcal{D}} |\text{Dist}(h(f(t_1), h(f(t_2))) - \text{Dist}(g(t_1), g(t_2))|. \quad (2)$$

This loss measures the difference in distances between randomly sampled text pairs in the converted and target spaces, encouraging global structure preservation. We use 1-cosine similarity as our distance metric. The second loss component, a local similarity loss, focuses on preserving neigh-



(a) Embeddings of 9 BEIR datasets (b) Embeddings across 2 models (c) Embeddings with converter

Figure 2: t-SNE visualization of embedding spaces across different corpora and models. (a) Embeddings of 9 diverse corpora from the BEIR datasets, highlighting the variability in embedding distributions across different datasets. (b) Comparison of gecko003 and gecko004 embeddings for the SciFact dataset, showcasing how different the embedding spaces between different model versions can be for the same dataset. (c) Embeddings of the gecko004 model and embeddings converted from gecko003 using the Embedding-Converter. The high degree of overlap indicates the successful alignment of embedding spaces achieved by the Embedding-Converter.

borhood relationships:

$$\mathcal{L}_{local} = \sum_{t_1 \in \mathcal{D}} \sum_{t_2 \in NN_k(t_1)} |\text{Dist}(h(f(t_1), h(f(t_2))) - \text{Dist}(g(t_1), g(t_2))|. \quad (3)$$

For each text t_1, this loss considers its k nearest neighbors ($(NN_k(t_1))$, based on target embedding similarities) and penalizes differences in relative distances between these neighbors in the converted and target spaces, thus preserving local neighborhoods (k is set to 100 in our experiments). The effect of these additional loss functions is evaluated empirically (Table 6). Ultimately, Embedding-Converter is jointly optimized using a weighted combination of all three losses:

$$h^* = \arg\min_h \mathcal{L}_{reg} + \alpha\mathcal{L}_{global} + \beta\mathcal{L}_{local}, \quad (4)$$

where $\alpha, \beta \geq 0$ are hyperparameters controlling the relative importance of each component, tuned using a validation set. All three losses are trained in batches for computational efficiency.

3.3 Implementation details

Embedding-Converter can be implemented with any architecture capable of mapping d_f-dimensional vectors to d_g-dimensional vectors. We primarily use a 4-layer perceptron with SELU activations (Klambauer et al., 2017). In Section 5, we show that a Transformer architecture (Vaswani, 2017) yields slightly lower performance. Model selection and hyperparameter optimization are driven by retrieval performance on a held-out validation set, aligning with the practical application of converted embeddings in retrieval tasks. Further hyperparameter and training details can be found in Appendix B.

4 Experiments

This section evaluates Embedding-Converter's performance across various scenarios. We begin by demonstrating its effectiveness in converting between different versions of the same model. We then assess its ability to bridge the gap between distinct model embedding spaces. While our primary focus is retrieval tasks, we also present results on other embedding-dependent tasks, such as text classification and semantic text similarity (STS) (Yang et al., 2018), to illustrate broader applicability. A detailed comparison of the computational time and cost of traditional re-embedding versus Embedding-Converter is provided in Appendix A.

4.1 Experimental settings

Embedding-Converter model is trained on a diverse set of text passages and queries from 14 datasets in the BEIR benchmark (Thakur et al., 2021). We use half the corpus for datasets with under 1 million passages, and 500,000 randomly sampled passages (e.g., 10% of Fever, Climate-fever, and HotPotQA) for larger datasets. To ensure sufficient query representation, the entire MSMarco query set (500K queries) is included in the training data. Consequently, MSMarco is excluded from in-domain

Dataset	gecko003 → gecko004			openai-3-small → gecko004		
	gecko003 (source)	gecko004 (target)	Embedding -Converter	openai-3-small (source)	gecko004 (target)	Embedding -Converter
Arguana	0.5189	0.6070	**0.6103**	0.5530	0.6070	**0.6049**
Climate-fever	0.2540	0.3369	**0.2959**	0.2792	0.3369	0.2716
DBPedia	0.4128	0.4677	**0.4322**	0.4154	0.4677	0.4099
Fever	0.7431	0.8106	**0.7786**	0.7227	0.8106	**0.7659**
FiQA	0.4582	0.5481	**0.5040**	0.4048	0.5481	**0.4393**
HotpotQA	0.6248	0.6892	0.5923	0.6121	0.6892	**0.6341**
NFCorpus	0.3284	0.3503	**0.3435**	0.3314	0.3503	**0.3479**
NQ	0.5166	0.6058	**0.5755**	0.5254	0.6058	**0.5653**
Quora	0.8626	0.8621	0.8392	0.8881	0.8621	0.8346
SciDocs	0.1836	0.2041	**0.1908**	0.2092	0.2041	0.1995
SciFact	0.7221	0.7693	**0.7601**	0.7292	0.7693	**0.7668**
Trec-covid	0.7454	0.7840	**0.8079**	0.8285	0.7840	**0.7983**
Touche	0.2161	0.2565	**0.2397**	0.2723	0.2565	**0.2706**
Average	0.5067	0.5609	**0.5362**	0.5209	0.5609	**0.5314**

Table 1: In-domain retrieval performance (in nDCG@10) of the Embedding-Converter on 13 BEIR datasets. Two conversion scenarios are presented: (i) intra-model conversion between different versions of Google's Gecko model (gecko003 to gecko004), and (ii) inter-model conversion from OpenAI's text-embedding-3-small model to Google's gecko004. **Bold** represents better performance than the source or target models.

evaluation to avoid bias.

We evaluate Embedding-Converter in two settings: in-domain and out-of-domain. In-domain performance is considered on the remaining 13 BEIR datasets using normalized Discounted Cumulative Gain at rank 10 (nDCG@10) (Järvelin and Kekäläinen, 2002). Out-of-domain generalization is considered on 12 held-out datasets from the CQADupStack benchmark (Hoogeveen et al., 2015), again using nDCG@10. Beyond retrieval, we evaluate Embedding-Converter on other embedding-dependent tasks, including text classification and STS, to demonstrate its broader applicability and assess the generalizability and transferability of the converted embeddings. Dataset details are in Appendix C.

4.2 Conversion between different model versions

To evaluate Embedding-Converter's ability to adapt to model updates, we use two versions of Google's Gecko text embedding model: gecko003 and gecko004[1]. We generate embeddings for our training data using both models and trained Embedding-Converter to map embeddings from gecko003 to gecko004 spaces. For evaluation, we converted the corpus embeddings of the 13 BEIR datasets and compared retrieval performance (nDCG@10) across three embedding sets: (1) original gecko003 embeddings, (2) original gecko004 embeddings, and (3) gecko003 embeddings converted to gecko004 space. Queries are consistently encoded using the target model (gecko004) to isolate the impact of corpus embedding conversion on retrieval effectiveness. For source/target model evaluation, we use the source/target model for both query and corpus embedding, respectively.

Table 1 shows Embedding-Converter's effectiveness. Converting from gecko003 to gecko004 significantly improves performance compared to using the original gecko003 embeddings. The converted embeddings' average performance falls between the source and target models for most datasets, and even approaches target model performance for some (e.g., Arguana, NFCorpus, and SciFact). This demonstrates Embedding-Converter's ability to efficiently transfer an entire corpus to a new embedding space with minimal performance loss. This makes it feasible to leverage newer models without the cost of re-embedding the entire corpus. As shown in Appendix A, Embedding-Converter offers significant cost and runtime savings (in the order of 100x) in real-world scenarios, with significant implications for maintaining and updating large-scale retrieval systems.

4.3 Conversion across different model families

To further demonstrate Embedding-Converter's versatility, we evaluate conversions between different embedding models. Specifically, we convert em-

[1] https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings

Dataset	gecko003 → gecko004			openai-3-small → gecko004		
	gecko003 (source)	gecko004 (target)	Embedding -Converter	openai-3-small (source)	gecko004 (target)	Embedding -Converter
Android	0.5258	0.5780	**0.5687**	0.5414	0.5780	**0.5576**
English	0.5019	0.5411	**0.5163**	0.5006	0.5411	**0.5017**
Gaming	0.6288	0.6720	**0.6422**	0.6125	0.6720	**0.6287**
Gis	0.3982	0.4503	**0.4223**	0.4055	0.4503	**0.4178**
Mathematica	0.2908	0.3621	**0.3329**	0.3053	0.3621	**0.3265**
Physics	0.4738	0.5291	**0.4981**	0.4615	0.5291	**0.4832**
Programmers	0.4455	0.5027	**0.4766**	0.4342	0.5027	**0.4627**
Stats	0.3531	0.4036	**0.3715**	0.3581	0.4036	**0.3644**
Tex	0.2958	0.3517	**0.3201**	0.2925	0.3517	**0.3018**
Unix	0.4362	0.4980	**0.4622**	0.4349	0.4980	**0.4498**
Webmasters	0.4297	0.4954	**0.4698**	0.4105	0.4954	**0.4466**
Wordpress	0.3453	0.3923	**0.3701**	0.3434	0.3923	**0.3493**
Average	0.4271	0.4814	**0.4542**	0.4250	0.4814	**0.4408**

Table 2: Out-of-domain retrieval performance (nDCG@10) of the Embedding-Converter on 12 CQADupStack datasets. Two conversion scenarios are presented: (i) intra-model conversion between different versions of Google's Gecko model (gecko003 to gecko004), and (ii) inter-model conversion from OpenAI's text-embedding-3-small model to Google's gecko004. **Bold** represents better performance than the source or target models.

beddings from OpenAI's text-embedding-3-small (openai-3-small)[2] to Google's gecko004. This is notable because these models have different embedding dimensions: 1536 for openai-3-small and 768 for gecko004. Using the same setup as before, a single Embedding-Converter was train to convert all corpora across the 13 BEIR datasets.

Table 1 shows that even with inter-model conversion and dimensionality reduction, Embedding-Converter significantly mitigates retrieval performance degradation. This has important practical implications, enabling efficient evaluation of new models on existing corpora without costly re-embedding. Specifically, Table 1(right) shows that the target model outperforms the source on 9 datasets, while the source is better on 4. Traditionally, determining the better model would require computing embeddings with both. However, Embedding-Converter offers a compelling alternative. By comparing performance with the source model, we can effectively approximate source/target comparisons without generating target embeddings. Our results confirm this – the relative performance of source and target models is accurately predicted by Embedding-Converter on 11 of the 13 datasets. This capability further highlights Embedding-Converter's value proposition. By facilitating seamless transitions between embedding spaces, it promotes flexibility, reduces computational overhead in developing and deploying embedding-based systems, and offers a valuable tool for preliminary model comparison.

[2] https://platform.openai.com/docs/guides/embeddings

4.4 Generalization to out-of-domain data

While the strong in-domain performance across 13 diverse datasets with a single Embedding-Converter is encouraging, evaluating out-of-domain generalization is crucial for practical use. Out-of-domain performance, particularly on unseen tasks with substantially different data distributions, is essential for assessing true generalizability. Correspondingly, we evaluate Embedding-Converter on 12 held-out CQADupStack datasets.

Table 2 shows the out-of-domain results. Even under these challenging conditions, Embedding-Converter consistently outperforms the source model, both for intra-model (gecko003 to gecko004) and inter-model (openai-3-small to gecko004) conversions. While the performance gap compared to the target model is larger than in the in-domain setting, Embedding-Converter still provides a valuable estimate of potential performance gains before re-embedding the entire corpus with the new model—a computationally expensive process. It offers a preliminary performance guarantee for migrating to a new model, enabling informed decisions about resource allocation. Notably, the relative performance of the source and target models is perfectly predicted by Embedding-Converter in this out-of-domain setting.

Task	Dataset	gecko003 → gecko004			openai-3-small → gecko004		
		gecko003 (source)	gecko004 (target)	Embedding -Converter	openai-3-small (source)	gecko004 (target)	Embedding -Converter
Classification	Toxic	0.9341	0.9446	**0.9392**	0.9380	0.9446	**0.9410**
	Tweet	0.7261	0.7535	**0.7425**	0.7476	0.7535	0.7434
	Average	0.8301	0.8491	**0.8409**	0.8428	0.8491	0.8422
STS	STS-13	0.7712	0.8047	**0.7982**	0.8425	0.8047	**0.8317**
	STS-14	0.7119	0.7403	**0.7359**	0.8001	0.7403	**0.7586**
	STS-22	0.7019	0.7246	**0.7080**	0.6716	0.7246	**0.6863**
	Average	0.7283	0.7565	**0.7474**	0.7714	0.7565	**0.7589**

Table 3: Classification and STS performances of Embedding-Converter in two different settings: (i) within same model lineup but different versions (gecko003 → gecko004), (ii) across different model lineup (openai-3-small → gecko004) on 5 datasets. **Bold** represents better performance than the source or target models.

4.5 Evaluate on other tasks beyond retrieval

While our primary focus has been retrieval, text embeddings are used in many applications. The MTEB benchmark (Muennighoff et al., 2022), with tasks like classification, clustering, reranking, and STS, highlights this versatility. To assess Embedding-Converter's broader applicability, we evaluate it on text classification (Toxic Conversation (cjadams, 2019) and Tweet Sentiment Extraction (Maggie, 2020) datasets) and semantic text similarity (STS-13 (Agirre et al., 2013), STS-14 (Bandhakavi et al., 2014), and STS-22 (Chen et al., 2022) datasets).

Table 3 shows the results. For the gecko003 to gecko004 conversion, the target model (gecko004) consistently outperforms the source (gecko003), and Embedding-Converter performs between the two, demonstrating its ability to transfer relevant embedding properties. For the openai-3-small to gecko004 conversion, the target model is better in 3 of 5 cases, while the source is better in the other 2. Importantly, Embedding-Converter correctly predicts the relative performance of the source and target models in 4 of the 5 cases, further demonstrating its utility for preliminary model comparison, even across different families.

Overall, these results suggest that the converted embeddings capture the target model's semantic information, enabling their effective use in diverse downstream tasks beyond retrieval. This generalization capability underscores Embedding-Converter's potential to facilitate efficient and flexible embedding model deployment across a wide range of applications, including unseen scenarios.

4.6 Leveraging for latency reduction

We've primarily focused on using Embedding-Converter to transform corpus embeddings, which is especially useful for large corpora. However, it also offers advantages when query latency is critical. Deploying large embedding models for online query processing is often impractical due to high latency. While corpus embeddings can be precomputed offline, query embeddings are generated in real time, creating a bottleneck. Developers may thus use smaller models for both queries and corpora, even if larger models would improve corpus representation and thus retrieval performance.

Conversion	Methods	In-domain	Out-domain
gecko003 → gecko004	Source model	0.5067	0.4271
	Target model	0.5609	0.4814
	Corpus converter	**0.5362**	**0.4542**
	Query converter	**0.5263**	**0.4348**
openai-3-small → gecko004	Source model	0.5209	0.4250
	Target model	0.5609	0.4814
	Corpus converter	**0.5314**	**0.4408**
	Query converter	0.5171	**0.4342**

Table 4: Embedding-Converter on query converting scenarios with two different settings: (i) within same model lineup but different versions (gecko003 → gecko004), (ii) across different model lineup (openai-3-small → gecko004). **Bold** represents better performance than the source or target models.

Embedding-Converter offers a solution by decoupling corpus and query embedding models. We can leverage larger models for corpus embeddings while maintaining low query latency by using a smaller model for initial query embedding generation and then converting these embeddings to the larger corpus model's space using Embedding-Converter. Table 4 shows the results of applying Embedding-Converter to queries instead of the cor-

Settings	Methods	gecko003 → gecko004		openai-3-small → gecko004	
		Global distance	Local distance	Global distance	Local distance
In-domain	Source model	0.1053	0.0246	0.2346	0.1260
	Converter	**0.0393**	**0.0163**	**0.0191**	**0.0205**
Out-domain	Source model	0.0805	0.0217	0.1811	0.1291
	Converter	**0.0325**	**0.0176**	**0.0179**	**0.0195**

Table 5: Comparison of global and local distance metrics (i.e., Eq. 2 and 3, lower the better for the Embedding-Converter on 13 BEIR and 12 CQADupStack datasets. Two conversion scenarios are presented: (i) intra-model conversion between different versions of Google's Gecko model (gecko003 to gecko004), and (ii) inter-model conversion from OpenAI's text-embedding-3-small model to Google's gecko004. **Bold** represents better performance.

pus. Query conversion achieves comparable performance to corpus conversion in most cases (with the exception of the in-domain conversion from openai-3-small). This demonstrates the potential of query conversion to improve retrieval performance in latency-constrained environments. By enabling the use of larger models for corpus embeddings without sacrificing query speed, Embedding-Converter provides a valuable tool for optimizing the accuracy-efficiency trade-off.

5 Discussions

5.1 Ablation studies

This section examines the impact of different loss functions and architectural choices on Embedding-Converter's performance. The model is trained with a combination of regression ($\mathcal{L}_{reg}$), global ($\mathcal{L}_{global}$), and local ($\mathcal{L}_{local}$) loss functions. We present ablation studies analyzing the effect of removing each loss component. We also explore architectural variations, comparing our default multi-layer perceptron (MLP) to a Transformer architecture. By systematically analyzing these modifications, we aim to identify the key factors contributing to Embedding-Converter's performance and understand their individual roles.

Table 6 summarizes our ablation study results, highlighting key factors influencing Embedding-Converter's performance:

- **Loss functions**: Both global and local loss functions are essential. Removing those degrades performance, particularly distance metrics, demonstrating their complementary roles.

- **Architecture variations**: The choice between Transformer and MLP matters, indicating sensitivity to architectural design, even with sufficient model capacity and proper training.

Variants	Performances		
	Global distance	Local distance	Retrieval
w/o $\mathcal{L}_{global}$ & $\mathcal{L}_{local}$	0.0452	0.0237	0.5219
Transformer	0.0233	0.0211	0.5273
Small networks (1/5x)	0.0203	0.0192	0.5263
Larger networks (5x)	**0.0177**	0.0164	0.5329
Only with MSMarco	0.0351	0.0227	0.5194
No variants	**0.0177**	**0.0163**	**0.5369**

Table 6: Ablation studies across different variants of Embedding-Converter. Global distance, local distance, and Retrieval performances are evaluated on out-domain retrieval tasks (with 12 CQADupStack datasets) - converting from gecko003 to gecko004.

- **Model size**: Smaller models perform slightly worse due to reduced capacity to capture complex relationships in embedding spaces. Larger models perform consistently with the original Embedding-Converter.

- **Data diversity**: Diverse training data significantly improves performance by enhancing generalization and coverage of the embedding space (Fig. 2(a)). Relying solely on MSMarco is insufficient for broad coverage.

5.2 Further analyses of Embedding-Converter performance

While evaluating Embedding-Converter on downstream tasks (Section 4) provides valuable insights, a complete assessment requires analyzing its ability to accurately align embedding spaces. We explore this alignment using quantitative, distance-based metrics, independent of specific downstream tasks.

We analyze both global and local distances between corpus embeddings. Global distances capture the overall structure and organization of the embedding space, while local distances focus on preserving relationships within local neighbor-

hoods. Analyzing both provides a comprehensive understanding of Embedding-Converter's effectiveness in accurately mapping embeddings between models while preserving the inherent structure of the embedding spaces.

Table 5 shows that Embedding-Converter effectively aligns both global and local distances, preserving meaningful positioning. This validates its ability to capture and replicate the structural properties of the target embedding space, further demonstrating its efficacy in cross-model mapping. Additional experiments, including reverse conversion, handling mixed embeddings, bridging open-source and black-box models, and multilingual embedding models are presented in Appendix D.

6 Conclusions

This paper addresses the critical challenge of embedding model incompatibility, a significant obstacle for practitioners navigating model updates and selection, and for the robustness of deployed systems. We introduce Embedding-Converter, a unified framework for efficiently translating between embedding models. Our design addresses the unique challenges of learning efficient embedding conversion through carefully chosen training mechanisms, demonstrating that end-to-end performance with converted embeddings can be largely preserved in many scenarios. This empowers practitioners with the flexibility to seamlessly transition between models, encouraging experimentation and facilitating the adoption of improved versions. Furthermore, Embedding-Converter can promote a paradigm shift in model development by encouraging the release of converters alongside new models, enabling a more user-centric approach to model migration. This fosters a more dynamic and user-friendly ecosystem for embedding models, prioritizing innovation and user experience, and ultimately contributing to a more robust and accessible environment for developing and deploying embedding-based applications.

7 Limitations and future works

While this paper demonstrates Embedding-Converter's ability to accurately convert embeddings across various models, we have not yet explored multimodal embeddings, leaving that for future work. Furthermore, converted embedding performance *does not fully match* that of the target model, meaning re-embedding is still necessary to achieve the full potential of a new model. However, Embedding-Converter provides an estimate of potential performance gains, allowing developers to invest in re-embedding with greater confidence.

References

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *SEM 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA. Association for Computational Linguistics.

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2017. Learning bilingual word embeddings with (almost) no bilingual data. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462.

Payal Bajaj, Daniel Campos, Nick Craswell, Li Deng, Jianfeng Gao, Xiaodong Liu, Rangan Majumder, Andrew McNamara, Bhaskar Mitra, Tri Nguyen, et al. 2016. Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

Anil Bandhakavi, Nirmalie Wiratunga, Deepak P, and Stewart Massie. 2014. Generating a word-emotion lexicon from #emotional tweets. In *Proceedings of the Third Joint Conference on Lexical and Computational Semantics (*SEM 2014)*, pages 12–21, Dublin, Ireland. Association for Computational Linguistics and Dublin City University.

Xi Chen, Ali Zeynali, Chico Camargo, Fabian Flöck, Devin Gaffney, Przemyslaw Grabowicz, Scott Hale, David Jurgens, and Mattia Samory. 2022. SemEval-2022 task 8: Multilingual news article similarity. In *Proceedings of the 16th International Workshop on Semantic Evaluation (SemEval-2022)*, pages 1094–1106, Seattle, United States. Association for Computational Linguistics.

inversion Jeffrey Sorensen Lucas Dixon Lucy Vasserman nithum cjadams, Daniel Borkan. 2019. Jigsaw unintended bias in toxicity classification.

Alexis Conneau, Guillaume Lample, Marc'Aurelio Ranzato, Ludovic Denoyer, and Hervé Jégou. 2017. Word translation without parallel data. *arXiv preprint arXiv:1710.04087*.

Doris Hoogeveen, Karin M Verspoor, and Timothy Baldwin. 2015. Cqadupstack: A benchmark data set for community question-answering research. In *Proceedings of the 20th Australasian document computing symposium*, pages 1–8.

Weihua Hu, Rajas Bansal, Kaidi Cao, Nikhil Rao, Karthik Subbian, and Jure Leskovec. 2022. Learning backward compatible embeddings. In *Proceedings of the 28th ACM SIGKDD Conference*

on Knowledge Discovery and Data Mining, pages 3018–3028.

Jui-Ting Huang, Ashish Sharma, Shuying Sun, Li Xia, David Zhang, Philip Pronin, Janani Padmanabhan, Giuseppe Ottaviano, and Linjun Yang. 2020. Embedding-based retrieval in facebook search. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2553–2561.

Florian Jaeckle, Fartash Faghri, Ali Farhadi, Oncel Tuzel, and Hadi Pouransari. 2023. Fastfill: Efficient compatible model update. *arXiv preprint arXiv:2303.04766*.

Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. 2017. Self-normalizing neural networks. *Advances in neural information processing systems*, 30.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, et al. 2022. Matryoshka representation learning. *Advances in Neural Information Processing Systems*, 35:30233–30249.

Chankyu Lee, Rajarshi Roy, Mengyao Xu, Jonathan Raiman, Mohammad Shoeybi, Bryan Catanzaro, and Wei Ping. 2024a. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Jinhyuk Lee, Feiyang Chen, Sahil Dua, Daniel Cer, Madhuri Shanbhogue, Iftekhar Naim, Gustavo Hernández Ábrego, Zhe Li, Kaifeng Chen, Henrique Schechter Vera, et al. 2025. Gemini embedding: Generalizable embeddings from gemini. *arXiv preprint arXiv:2503.07891*.

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

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

Wei Chen Maggie, Phil Culliton. 2020. Tweet sentiment extraction.

Marvin Marcus. 1971. Linear transformations on matrices. *J. Res. Nat. Bur. Standards Sect. B*, 75:107–113.

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

Arvind Neelakantan, Tao Xu, Raul Puri, Alec Radford, Jesse Michael Han, Jerry Tworek, Qiming Yuan, Nikolas Tezak, Jong Wook Kim, Chris Hallacy, et al. 2022. Text and code embeddings by contrastive pretraining. *arXiv preprint arXiv:2201.10005*.

Jianmo Ni, Chen Qu, Jing Lu, Zhuyun Dai, Gustavo Hernández Ábrego, Ji Ma, Vincent Y Zhao, Yi Luan, Keith B Hall, Ming-Wei Chang, et al. 2021. Large dual encoders are generalizable retrievers. *arXiv preprint arXiv:2112.07899*.

Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. 2019. Relational knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3967–3976.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR.

Vivek Ramanujan, Pavan Kumar Anasosalu Vasu, Ali Farhadi, Oncel Tuzel, and Hadi Pouransari. 2022. Forward compatible training for large-scale embedding retrieval systems. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 19386–19395.

Shafiq Rayhan Joty Caiming Xiong Yingbo Zhou Semih Yavuz Rui Meng, Ye Liu. 2024. Sfr-embedding-mistral:enhance text retrieval with transfer learning. Salesforce AI Research Blog.

Tim Schopf, Dennis N Schneider, and Florian Matthes. 2023. Efficient domain adaptation of sentence embeddings using adapters. *arXiv preprint arXiv:2307.03104*.

Yantao Shen, Yuanjun Xiong, Wei Xia, and Stefano Soatto. 2020. Towards backward-compatible representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6368–6377.

Thomas S Shores et al. 2007. *Applied linear algebra and matrix analysis*, volume 2541. Springer.

Nandan Thakur, Nils Reimers, Andreas Rücklé, Abhishek Srivastava, and Iryna Gurevych. 2021. Beir: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*.

François Treves. 2013. *Topological vector spaces, distributions and kernels*. Courier Corporation.

A Vaswani. 2017. Attention is all you need. *Advances in Neural Information Processing Systems*.

Kexin Wang, Nandan Thakur, Nils Reimers, and Iryna Gurevych. 2021. Gpl: Generative pseudo labeling for unsupervised domain adaptation of dense retrieval.

Liang Wang, Nan Yang, Xiaolong Huang, Binxing Jiao, Linjun Yang, Daxin Jiang, Rangan Majumder, and Furu Wei. 2022. Text embeddings by weakly-supervised contrastive pre-training. *arXiv preprint arXiv:2212.03533*.

Suhang Wang, Jiliang Tang, Charu Aggarwal, and Huan Liu. 2016. Linked document embedding for classification. In *Proceedings of the 25th ACM international on conference on information and knowledge management*, pages 115–124.

Yinfei Yang, Steve Yuan, Daniel Cer, Sheng-yi Kong, Noah Constant, Petr Pilar, Heming Ge, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Learning semantic textual similarity from conversations. *arXiv preprint arXiv:1804.07754*.

Jinsung Yoon, Yanfei Chen, Sercan Arik, and Tomas Pfister. 2024. Search-adaptor: Embedding customization for information retrieval. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12230–12247.

Jiahui Yu, Zirui Wang, Vijay Vasudevan, Legg Yeung, Mojtaba Seyedhosseini, and Yonghui Wu. 2022. Coca: Contrastive captioners are image-text foundation models. *arXiv preprint arXiv:2205.01917*.

Andrew Zhai, Hao-Yu Wu, Eric Tzeng, Dong Huk Park, and Charles Rosenberg. 2019. Learning a unified embedding for visual search at pinterest. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2412–2420.

Xinyu Zhang, Nandan Thakur, Odunayo Ogundepo, Ehsan Kamalloo, David Alfonso-Hermelo, Xiaoguang Li, Qun Liu, Mehdi Rezagholizadeh, and Jimmy Lin. 2023. Miracl: A multilingual retrieval dataset covering 18 diverse languages. *Transactions of the Association for Computational Linguistics*, 11:1114–1131.

A Computational complexity

We analyze the computational cost and processing time of Embedding-Converter across various scenarios to quantify its benefits. Assuming a document length of 256 tokens (on average), we used three embedding models with different pricing and request-per-minute (RPM) limits. This analysis provides a concrete comparison of Embedding-Converter's efficiency gains against traditional corpus re-embedding.

- **Openai-3-large**: Price: \$0.065 / 1M tokens [3] & RPM: 1M tokens [4] (with free tier)
- **Openai-3-small**: Price: \$0.010 / 1M tokens & RPM: 1M tokens (with free tier)
- **Google's Gecko004**: Price: \$0.00002 / 1K characters [5] & RPM: 7500 inputs [6]

Embedding model	Corpus size	Computational cost		Computational time	
		Baseline	Embedding -Converter	Baseline	Embedding -Converter
Openai-3-large	1B 50M	\$16640 \$832	\$185 \$10	4266 hours 213 hours	37 hours 1.9 hours
Openai-3-small	1B 50M	\$2560 \$128	\$115 \$6	4266 hours 213 hours	23 hours 1.2 hours
gecko004	1B 50M	\$20480 \$1024	\$75 \$4	2222 hours 111 hours	15 hours 0.8 hours

Table 7: Computational cost and time comparisons with Embedding-Converter for different corpus sizes. The baseline represents re-embedding the entire corpus using the respective embedding models.

While OpenAI's higher-tier API offers increased RPM, potentially reducing computation time, it's still significantly slower than Embedding-Converter. Moreover, OpenAI's cost per API call remains constant across tiers, offering no cost advantage for higher RPM usage. Embedding-Converter, even without low-level optimizations, achieves remarkable efficiency with modest compute requirements. It can process a 50 million document corpus in under two hours (including data loading), and inference alone with a pre-trained converter takes only 20 minutes with openai-3-small. This is over 100x faster than traditional re-embedding.

Using 2 V100 GPUs, Embedding-Converter costs \$4.96 per hour on Google Cloud[7]. This translates to a cost reduction exceeding 100x compared to directly generating target model embeddings. These results highlight Embedding-Converter's substantial efficiency gains, providing a compelling solution for migrating to new embedding models with significant cost and time savings, particularly for large corpora.

B Hyper-parameters & training details

We implement Embedding-Converter using a 4-layer multi-layer perceptron (MLP) with hidden layer dimensions of (5x output dimension, 5x output dimension, 5x output dimension, output dimension). For the gecko003 to gecko004 conversion, this resulted in a 35 million parameter model. SELU activations and L2 normalization on the output are used. The Adam optimizer with a learning rate of 0.001 is used for training, which proceeded for 50,000 iterations. To mitigate dataset bias and improve coverage, batches of 64 are sampled uniformly from each of the 14 BEIR datasets, ensuring equal representation. Validation performance (using the Scifact dataset, comprising 1109 queries, 1258 labels, and 5183 corpus passages) is evaluated every 250 iterations, and the best performing model is selected. The global and local loss weight hyperparameters α and β are tuned within the range of $[0.01, 1.0]$, with $\alpha = \beta = 0.1$ generally providing good results. The neighborhood size (k) for the local distance loss is set to 100 across all experiments.

[3] https://openai.com/api/pricing/
[4] https://platform.openai.com/docs/guides/rate-limits/usage-tiers?context=tier-free
[5] https://cloud.google.com/vertex-ai/generative-ai/pricing
[6] https://cloud.google.com/vertex-ai/generative-ai/docs/quotas#text-embedding-limits
[7] https://cloud.google.com/compute/gpus-pricing

C Data statistics

C.1 BEIR datasets

Datasets	Number of queries	Number of test pairs	Number of corpus
Arguana	1406	1406	8674
Climate-fever	1535	4681	5416593
DBPedia	467	49188	4635922
Fever	123142	148022	5416568
FiQA	6648	15872	57638
HotPotQA	97852	184810	5233329
NFCorpus	3237	122909	3633
NQ	3452	4201	2681468
Quora	15000	23301	522931
SciDocs	1000	29928	25657
SciFact	1109	1258	5183
Trec-Covid	50	66336	171332
Touche	49	2214	382545

Table 8: The statistics of 13 BEIR datasets (sorted by the alphabetical order).

C.2 CQADupStack datasets

Datasets	Number of queries	Number of test pairs	Number of corpus
Android	699	1696	22998
English	1570	3765	40221
Gaming	1595	2263	45301
Gis	885	1114	37637
Mathematica	804	1358	16705
Physics	1039	1933	38316
Programmers	876	1675	32176
Stats	652	913	42269
Tex	2906	5154	68184
Unix	1072	1693	47382
Webmasters	506	1395	17405
Wordpress	541	744	48605

Table 9: The statistics of 12 CQADupStack datasets (sorted by alphabetical order). We only use test samples for the evaluation.

C.3 STS and classification datasets

Tasks	Datasets	Number of train samples	Number of test samples	Number of classes
Classification	Toxic	50000	50000	2
	Tweet	27481	3534	3
STS	STS-13	-	1500	-
	STS-14	-	3750	-
	STS-22	-	197	-

Table 10: The statistics of 2 classification and 3 STS datasets. We only use test samples for the evaluation.

D Additional experiments

D.1 Converting from new to old embedding models

To further demonstrate Embedding-Converter's versatility, we conduct experiments converting from a newer to an older model. This "downgrading" scenario might occur due to resource constraints or compatibility needs. Specifically, we reverse the conversion direction from gecko004 (source) to gecko003 (target), contrasting with the conversions shown in Table 1 (left) and Table 2 (left).

Dataset	gecko004 → gecko003			gecko004 → openai-3-small		
	gecko004 (source)	gecko003 (target)	Embedding -Converter	gecko004 (source)	openai-3-small (target)	Embedding -Converter
Arguana	0.6070	0.5189	0.5148	0.6070	0.5530	0.5713
Climate-fever	0.3369	0.2540	0.2905	0.3369	0.2792	0.2931
DBPedia	0.4677	0.4128	0.3979	0.4677	0.4154	0.3898
Fever	0.8106	0.7431	0.7327	0.8106	0.7227	0.6972
FiQA	0.5481	0.4582	0.4824	0.5481	0.4048	0.4507
HotpotQA	0.6892	0.6248	0.5794	0.6892	0.6121	0.5519
NFCorpus	0.3503	0.3284	0.3347	0.3503	0.3314	0.3318
NQ	0.6058	0.5166	0.5147	0.6058	0.5254	0.5151
Quora	0.8621	0.8626	0.8369	0.8621	0.8881	0.8396
SciDocs	0.2041	0.1836	0.1743	0.2041	0.2092	0.1928
SciFact	0.7693	0.7221	0.7227	0.7693	0.7292	0.7074
Trec-covid	0.7840	0.7454	0.7187	0.7840	0.8285	0.8278
Touche	0.2565	0.2161	0.2423	0.2565	0.2723	0.2684
Average	0.5609	0.5067	0.5032	0.5609	0.5209	0.5105

Table 11: In-domain retrieval performance (in nDCG@10) of the Embedding-Converter on 13 BEIR datasets. Two conversion scenarios are presented: (i) intra-model conversion between different versions of Google's Gecko model (gecko004 to gecko003), and (ii) inter-model conversion from Google's gecko004 to OpenAI's text-embedding-3-small model.

The results, presented in Table 11 (left) and Table 12 (left), show that Embedding-Converter's performance remains remarkably consistent with the (older) target model. This highlights the flexibility of our approach, supporting both upgrading and downgrading of embedding models for a wider range of practical applications.

D.2 Converting from smaller dimensional embedding to larger dimensional embeddings

While the main manuscript focuses on conversions where the target model has equal or smaller dimensionality, we also explore the challenging, yet practical, scenario where the target model has higher dimensionality, as is often the case with newer models. Specifically, we train and evaluate Embedding-Converter with gecko004 (768 dimensions) as the source and openai-3-small (1536 dimensions) as the target.

Dataset	gecko004 → gecko003			gecko004 → openai-3-small		
	gecko004 (source)	gecko003 (target)	Embedding -Converter	gecko004 (source)	openai-3-small (target)	Embedding -Converter
Android	0.5780	0.5258	0.5172	0.5780	0.5414	0.5374
English	0.5411	0.5019	0.4785	0.5411	0.5006	0.4844
Gaming	0.6720	0.6288	0.6175	0.6720	0.6125	0.6052
Gis	0.4503	0.3982	0.4008	0.4503	0.4055	0.3951
Mathematica	0.3621	0.2908	0.2879	0.3621	0.3053	0.2984
Physics	0.5291	0.4738	0.4750	0.5291	0.4615	0.4670
Programmers	0.5027	0.4455	0.4479	0.5027	0.4342	0.4460
Stats	0.4036	0.3531	0.3444	0.4036	0.3581	0.3384
Tex	0.3517	0.2958	0.2849	0.3517	0.2925	0.2879
Unix	0.4980	0.4362	0.4287	0.4980	0.4349	0.4329
Webmasters	0.4954	0.4297	0.4345	0.4954	0.4105	0.4338
Wordpress	0.3923	0.3453	0.3289	0.3923	0.3434	0.3334
Average	0.4814	0.4271	0.4205	0.4814	0.4250	0.4217

Table 12: Out-of-domain retrieval performance (nDCG@10) of the Embedding-Converter on 12 CQADupStack datasets. Two conversion scenarios are presented: (i) intra-model conversion between different versions of Google's Gecko model (gecko004 to gecko003), and (ii) inter-model conversion from Google's gecko004 to OpenAI's text-embedding-3-small model.

The results (Table 11 (right) and Table 12 (right)) show that our method handles this conversion with minimal performance degradation. This further demonstrates Embedding-Converter's robustness and generalizability, showing its ability to effectively bridge embedding spaces even when the target dimensionality is greater than the source.

D.3 Embedding-converter with mixed embeddings

Real-world applications often involve dynamically updating corpora. Embedding-Converter offers a significant advantage here as well. Instead of converting new documents to the source embedding space before target embedding generation, we can directly embed them using the target model, resulting in a mixed corpus of converted (older documents) and new embeddings. To simulate this, we randomly replace half of the corpus embeddings with target embeddings.

Dataset	gecko003 → gecko004			
	gecko003	gecko004	Embedding-Converter	
	(source)	(target)	Standard	Mixed
Arguana	0.5189	0.6070	0.6103	0.6082
Climate-fever	0.2540	0.3369	0.2959	0.3124
DBPedia	0.4128	0.4677	0.4322	0.4486
Fever	0.7431	0.8106	0.7786	0.7946
FiQA	0.4582	0.5481	0.5040	0.5196
HotpotQA	0.6248	0.6892	0.5923	0.6410
NFCorpus	0.3284	0.3503	0.3435	0.3466
NQ	0.5166	0.6058	0.5755	0.5435
Quora	0.8626	0.8621	0.8392	0.8304
SciDocs	0.1836	0.2041	0.1908	0.1963
SciFact	0.7221	0.7693	0.7601	0.7671
Trec-covid	0.7454	0.7840	0.8079	0.7865
Touche	0.2161	0.2565	0.2397	0.2481
Average	0.5067	0.5609	0.5362	0.5419

Table 13: In-domain retrieval performance (in nDCG@10) of the Embedding-Converter on 13 BEIR datasets from gecko003 to gecko004. Two conversion scenarios are presented: (i) Standard: with 100% converted corpus, (ii) Mixed: with 50% converted corpus and 50% target corpus.

The results (Table 13 and 14) show that performance in this mixed setting exceeds the scenario where all embeddings are converted. This highlights two key strengths: (1) Compatibility: Converted embeddings seamlessly integrate with new embeddings, demonstrating strong space compatibility. (2) Generalizability: Embedding-Converter effectively handles mixed embeddings, further validating its robustness and real-world applicability.

Dataset	gecko003 → gecko004			
	gecko003	gecko004	Embedding-Converter	
	(source)	(target)	Standard	Mixed
Android	0.5258	0.5780	0.5687	0.5632
English	0.5019	0.5411	0.5163	0.5255
Gaming	0.6288	0.6720	0.6422	0.6547
Gis	0.3982	0.4503	0.4223	0.4394
Mathematica	0.2908	0.3621	0.3329	0.3490
Physics	0.4738	0.5291	0.4981	0.5148
Programmers	0.4455	0.5027	0.4766	0.4877
Stats	0.3531	0.4036	0.3715	0.3846
Tex	0.2958	0.3517	0.3201	0.3323
Unix	0.4362	0.4980	0.4622	0.4775
Webmasters	0.4297	0.4954	0.4698	0.4781
Wordpress	0.3453	0.3923	0.3701	0.3807
Average	0.4271	0.4814	0.4542	0.4656

Table 14: Out-of-domain retrieval performance (nDCG@10) of the Embedding-Converter on 12 CQADupStack datasets from gecko003 to gecko004. Two conversion scenarios are presented: (i) Standard: with 100% converted corpus, (ii) Mixed: with 50% converted corpus and 50% target corpus.

D.4 Embedding-converter with multiple versions of embedding models

Embedding models are constantly updated, creating the challenge of converting embeddings across multiple versions. For example, a user might need to transition from gecko003 to GTE-Large and then to gecko004. While a direct gecko003 to gecko004 conversion is possible, we also explored using a sequence of converters: gecko003 to GTE-Large, then GTE-Large to gecko004. This sequential approach might be beneficial when direct conversion is costly or when intermediate embeddings are needed.

Dataset	gecko003 → gecko004			
	gecko003	gecko004	Embedding-Converter	
	(source)	(target)	Direct	Multiple
Arguana	0.5189	0.6070	0.6103	0.5812
FiQA	0.4582	0.5481	0.5040	0.4903
NFCorpus	0.3284	0.3503	0.3435	0.3470
Quora	0.8626	0.8621	0.8392	0.8361
SciDocs	0.1836	0.2041	0.1908	0.1953
SciFact	0.7221	0.7693	0.7601	0.7626
Trec-covid	0.7454	0.7840	0.8079	0.7580
Touche	0.2161	0.2565	0.2397	0.2358
Average	0.5044	0.5609	0.5369	0.5258

Table 15: In-domain retrieval performance (in nDCG@10) of the Embedding-Converter on 8 BEIR datasets from gecko002 to gecko004. Two conversion scenarios are presented: (i) Direct: converting gecko003 to gecko004 directly, (ii) Multiple: converting gecko003 to GTE-Large first and then converting GTE-Large to gecko004.

Table 15 compares these strategies. Direct conversion performed slightly better, but the difference was minimal. This demonstrates Embedding-Converter's flexibility and its ability to effectively handle multi-version conversions, providing a practical solution for navigating the evolving embedding model landscape.

D.5 Converting open-source model to black-box model

To further demonstrate Embedding-Converter's versatility, we evaluate conversions between open-source and black-box embedding models, a crucial aspect for ensuring compatibility and facilitating transitions across different model ecosystems. Specifically, we convert embeddings from the open-source GTE-Large model (Li et al., 2023) to Google's black-box gecko004 model.

Dataset	GTE-Large → gecko004		
	GTE-Large (source)	gecko004 (target)	Embedding -Converter
Arguana	0.5928	0.6070	0.6081
FiQA	0.4434	0.5481	0.5059
NFCorpus	0.3391	0.3503	0.3478
Quora	0.8824	0.8621	0.8391
SciDocs	0.2330	0.2041	0.2080
SciFact	0.7402	0.7693	0.7689
Trec-covid	0.7053	0.7840	0.7628
Touche	0.2237	0.2565	0.2431
Average	0.5200	0.5477	0.5355

Table 16: In-domain retrieval performance (in nDCG@10) of the Embedding-Converter on 8 BEIR datasets across inter-model conversion from GTE-Large to Google's gecko004 model.

Table 16 shows that Embedding-Converter effectively bridges these models across various BEIR datasets, maintaining strong performance. This highlights the generalizability of our approach and its ability to handle diverse conversion scenarios, including those involving both open-source and proprietary models.

D.6 Embedding-Converter on multilingual datasets

In this subsection, we leveraged the well-known MIRACL datasets (Zhang et al., 2023), to evaluate the generalizability of Embedding-Converter across different languages. Here, we utilized textembedding-gecko-multilingual@001 as the source embedding model and text-multilingual-embedding-002[8] as the target embedding model.

Dataset	gecko-multilingual-001 → gecko-multilingual-002		
	gecko-multilingual-001 (source)	gecko-multilingual-002 (target)	Embedding-Converter
Swahili	0.6745	0.6660	0.6749
Telugu	0.7536	0.7540	0.7576
Thai	0.6605	0.6641	0.6614
Chinese	0.5072	0.5322	0.5126
Japanese	0.5551	0.5339	0.5379
Russian	0.5114	0.5446	0.5272
French	0.4098	0.4924	0.4634
Germany	0.4295	0.4928	0.4742

Table 17: Multilingual retrieval performance (in nDCG@10) of the Embedding-Converter on 8 MIRACL datasets across intra-model conversion from Google's gecko-multilingual-001 to gecko-multilingual-002 models.

Table 17 clearly demonstrates the ability of Embedding-Converter to successfully transform source embeddings into target embeddings within multilingual contexts.

D.7 Embedding-Converter with smaller dimensions

In this subsection, we further verify the generalizability of Embedding-Converter on smaller dimensional embeddings because techniques like Matryoshka embedding (Kusupati et al., 2022) can integrate to reduce embedding dimension via truncation. Since OpenAI's openai-3-small model possesses Matryoshka properties, we designed an experiment to evaluate our converter in this context. Specifically, we trained the Embedding-Converter using 384-dimensional truncated versions of openai-3-small to original gecko004 model (with 768 dimensions) as source and target embeddings, to highlight the strength of the Embedding-Converter for low-to-high dimensional transformation settings.

Dataset	openai-3-small-384-truncated → gecko-004		
	openai-3-small-384-truncated (source)	gecko-004 (target)	Embedding-Converter
Arguana	0.5445	0.6070	0.5862
FiQA	0.3633	0.5481	0.4140
NFCorpus	0.3164	0.3503	0.3428
Quora	0.8816	0.8621	0.8330
SciDocs	0.1923	0.2041	0.1935
SciFact	0.7105	0.7693	0.7516
Trec-covid	0.7984	0.7840	0.7813
Touche	0.2634	0.2565	0.2484

Table 18: Retrieval performance (in nDCG@10) of the Embedding-Converter on 8 BEIR datasets across intra-model conversion from 384-dimensional truncated versions of openai-3-small to Google's gecko-004 models (from 384 dimensions to 768 dimensions).

Table 18 demonstrates that our Embedding-Converter consistently outperforms the original source embeddings, even when applied to the lower dimensional transformation settings.

[8] https://cloud.google.com/vertex-ai/generative-ai/docs/embeddings/get-text-embeddings

D.8 Embedding-Converter from BERT-based model to decoder-only model

In this subsection, we introduce additional experiments with the following experimental setup. Given that the Gemini embedding model(Lee et al., 2025) represents the current state-of-the-art, particularly for decoder-only architectures, we designed our experiment to explore the potential of converting from a BERT based embedding model to this advanced model. Specifically, we utilize GTE-Large as our source model and the Gemini embedding as our target model. We then train the Embedding-Converter to map embeddings from the source to the target. The performance of this converted embedding is subsequently evaluated on the subsets of BEIR benchmark datasets.

Dataset	GTE-Large → Gemini embedding		
	GTE-Large (source)	Gemini embedding (target)	Embedding-Converter
Arguana	0.5928	0.6539	0.6469
NFCorpus	0.3391	0.4131	0.3923
SciDocs	0.2330	0.2396	0.2280
SciFact	0.7402	0.9680	0.9218

Table 19: Retrieval performance (in nDCG@10) of the Embedding-Converter on 4 BEIR datasets across inter-model conversion from GTE-Large to Google's Gemini embedding models (from BERT-based model to decoder-only model).

Table 19 demonstrates that, irrespective of the underlying architecture of the embedding models involved, our proposed Embedding-Converter effectively transforms the source embeddings into representations more closely aligned with the target Gemini embedding.