Integrating Vector Databases across Embedding Models

BEINING YANG, University of Edinburgh, UK
YANG CAO, University of Edinburgh, UK
YANG REN, Edinburgh Research Center, Central Software Institute, Huawei, UK

Vector databases have been widely used to implement similarity search over unstructured objects, *e.g.,* documents and images. Each vector database is produced by an embedding model that encodes the objects in a way such that more similar objects are embedded to closer vectors, allowing us to use top-k vector search as an implementation of top-k object similarity search. It is common practice that different vector databases use distinct embedding models and the same object may be encoded by different embedding vectors across databases. As a result, one cannot share and integrate vector databases to expand similarity search across datasets, a property we take for granted for relational databases. In this work, we attempt to break the barrier between different vector databases, by developing an approach to integrating vector databases generated by different embedding models, with neither any access to the encoded data objects nor knowledge of the embedding models. Our approach is rooted in the *local isometry hypothesis*, a finding made via extensive experiments on real-life embedding vectors, and is backed up by theoretical analysis that bounds the quality of integrated vector database. Experimental results show that we can integrate vector databases produced by various popular embedding models, *e.g.,* NV-embed-V2, OpenAI Ada, GloVe, Mistral and FastText, while offering high recall of top-k similarity search over the integrated datasets.

CCS Concepts: • **Information systems → Retrieval models and ranking**; **Information integration**; • **Computing methodologies → Machine learning**.

Additional Key Words and Phrases: Vector databases; Data integration; Cross-model vector database integration

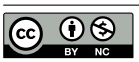
ACM Reference Format:
Beining Yang, Yang Cao, and Yang Ren. 2025. Integrating Vector Databases across Embedding Models. *Proc. ACM Manag. Data* 3, 6 (SIGMOD), Article 338 (December 2025), 28 pages. https://doi.org/10.1145/3769803

1 Introduction

A vector database is a vector representation of a collection of unstructured data items, *e.g.,* text and images, such that more similar items are encoded by vectors that are closer in the vector space. This allows us to use vector search over the encoded vectors to implement similarity search over unstructured data. The mapping from data items to vectors is done via an embedding model, which is often part of a neural network that is capable of accurately capturing the extent of dissimilarity between data items via distances between the corresponding vectors that encoding them.

Advancements in embedding models have made vector search highly effective for text [14, 50, 65, 77, 92] and image [33, 34, 42, 81] retrival. This has led to widespread adoption of vector databases in applications like recommender systems [52, 75, 89, 90] and retrieval-augmented generation (RAG) for LLMs [22, 25, 73], where searches run constantly over curated embedding vectors.

Authors' Contact Information: Beining Yang, B.Yang-32@sms.ed.ac.uk, University of Edinburgh, Edinburgh, UK; Yang Cao, yang.cao@ed.ac.uk, University of Edinburgh, Edinburgh, UK; Yang Ren, Edinburgh Research Center, Central Software Institute, Huawei, Edinburgh, UK, renyang1@huawei.com.



ACM 2836-6573/2025/12-ART338
https://doi.org/10.1145/3769803

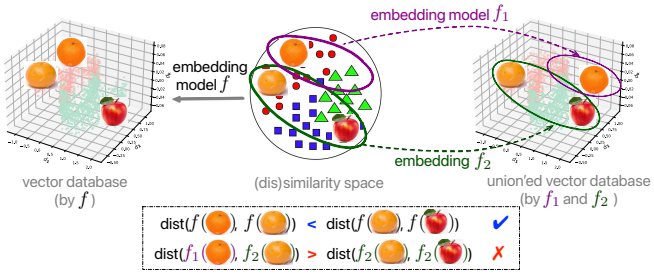


Fig. 1. What happens when we union two vector databases? (The union of vector databases embedded by f_1 and f_2 is not a vector database of the dataset that would be embedded by f.)

Vector databases cannot share data. Although vector databases are a widely accepted terminology, they lack one of the fundamental properties of traditional databases – data sharing. Sharing data in the form of a database across different parties has been an instrumental property taken for granted by relational database system users. The property was even explicitly highlighted by System R, the first database system [6] ever built. This allows us to collect and integrate data from multiple sources databases [23, 28, 35, 64], and query the integrated data for a complete answer that covers all sources.

However, it is not so straightforward to share vector databases as they each is "encrypted" by the embedding model that generates the vectors. The same data item is thus encoded differently across vector databases generated by different embedding models. As a result, it does not make any sense to union two vector databases, although this would enable us to share and integrate vector databases from different embedding models of varying data sources. As illustrated in Fig. 1, simply taking the union of two vector databases embedded by different models would render vector search useless for similarity search over the integrated dataset, as it breaks the fundamental principle that more similar items should be encoded by closer vectors.

Indeed, it is commonplace that different vector databases are embedded by distinct, customized and fine-tuned embedding models. As pointed out by major industry vendors, finetuning embedding models on in-domain data can significantly improve vector search and RAG accuracy, and significant industrial effort has been devoted to make customizing embedding models efficient and accessible to public users [20]. In addition, a substantial part of unstructured data are generated on edge devices which use customized device-specific embedding models [15]. These diverse embedding models produce a vast number of isolated, non-exchangeable vector databases.

Our vision. We aim to realize the vision of sharing and integrating vector databases from *different* embedding models, *even if we have access only to the vectors and are oblivious to the embedding models*. While data integration has long been a classic topic in our community and has been studied for decades, it primarily deals with relational databases and does not help with vector databases, which lack database schemas and use only numeric values that carry no explicit semantics.

Impact. We envisage that realizing this vision could benefit a range of directions, from practical industrial settings to data sharing scenarios where explicit record sharing is either prohibited or inefficient, yet there is still a need for collective searches and analyses. Below are a few examples.

(A1) *Embedding backfilling*. Aligning and integrating vectors across embedding models can help with the embedding backfilling problem. As noted in [38], in industrial settings, the embedding model is constantly updated, resulting in embedding vectors that are incompatible with historical records over time. Moreover, historical versions of the embeddings can never be retired as they remain used by legacy applications. As such, we need a system-wide overhaul to backfill vectors and unify with all applications. Our vision could potentially help with this emerging industrial

challenge [1], by making embedding vectors "talk" with each other without backfills.

(A2) *Privacy-preserving federated image/document search.* Privacy-preserving searching over unstructured datasets such as images and documents managed by autonomous parties is challenging, if not impossible at all. While privacy-preserving federated learning [45] offers various privacy controls, it enforces all parties to use the same learned embedding model. Realizing our vision would allow each autonomous party to use its own embedding model while still offering search over the entire collection of datasets from all parties, without actually moving or revealing their data records.

(A3) *Cloud-edge similarity search.* It is common to have data residing in both edge devices and the cloud, each independently embedded by different embedding models. A collaborative search involves finding relevant *e.g.,* images in the cloud that are similar to a query image on an edge device. Sending the query image to the cloud would incur heavy communication costs and may not be feasible due to privacy concerns. By bridging and integrating on-device embedding vectors with those in the cloud, one would only need to send the query embedding vector for retrieval, which is more secure and cost-efficient than sending the query image directly.

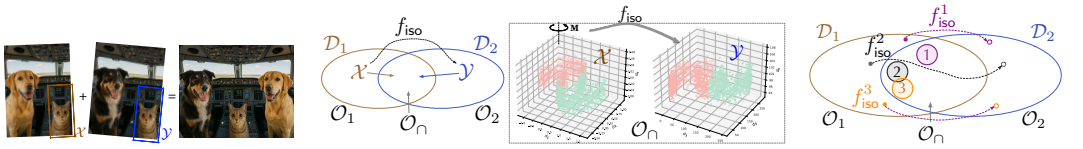
Prior work & challenges. Realizing the vision is challenging. Previous work on vector databases and similarity search mostly focuses on efficiency by designing vector indices [7, 26, 60, 61] and building dedicated vector database systems [11, 32, 67, 83, 86]. They offer little help in dealing with the interoperability between vector databases from different embedding models. As an added challenge, we aim to integrate vector databases by examining the embedding vectors only. This implies that we cannot reconstruct a global vector database by re-embedding data items, nor train a model to learn a mapping between the embedding models from the dataset.

Towards vector database integration. As a first attempt to realize this vision, we present a data-driven approach to aligning and integrating vector databases generated by different embedding models. It is based on a simple hypothesis: if two embedding models capture the same similarity over the same data items, their geometric shapes in the high-dimensional vector space should be approximately isometric. However, via extensive experiments over real-life embedding vectors, we find that the hypothesis does not hold globally but applies to local regions of the vector database where distances between vectors are relatively short. We refer to this phenomenon as the *local isometry hypothesis* and verify its existence in benchmarks.

The hypothesis gives us a method to combine vector databases across models. The idea is to use vectors that encode the same data items across the two vector databases as references. Such references can be obtained via *e.g.,* known correspondence between different embedding versions in backfilling scenarios, data linkage in datasets, or by querying remote embedding models via sampled vectors. We then compute a collection of *local isometries* that align the references across databases, and use them to integrate new vectors that are not references.

We find that the effectiveness of local isometries for transforming and integrating new embedding vectors is strongly connected to their scopes and determining the scopes of local isometries for integrating different vector databases is a central problem underpinning the approach. To tackle this, we prove an upper bound of the integration error of local isometries, which connects the scope of isometries to the singular values, intrinsic dimensions and conditioning of the vector database viewed as a matrix of column vectors. Such connection gives us guidelines for determining the best scopes of isometries of a given pair of vector databases. Using benchmarks, we show that our method can integrate vector databases across major embedding models, *e.g.,* OpenAI Ada, GloVe, Mistral, NV-embed-V2 and FastText, while offering high recall of top-k search over the integrated data.

Organization. In Section 2, we propose the idea of integrating vector databases and present the initial isometry hypothesis. We evaluate the hypothesis in Section 3 and consolidate it into the



(a) Image fusion (b) Hypothesis A: globally isometric (c) Hypothesis B: locally isometric

Fig. 2. Drawing parallel between image correlation in computer vision (a) and vector database integration (b-c): (a) fuses two images into one by taking the cat as reference to compute a rigid motion ($\mathcal{X} \to \mathcal{Y}$) that rotates, reflects, scales and translates the first image and fuses it with the second. (b) Hypothesis A assumes *global consistency* between between $\mathcal{D}_1[\mathcal{O}_\cap]$ and $\mathcal{D}_2[\mathcal{O}_\cap]$; by treating $\mathcal{O}_\cap$ as the cat in (a), we determine an isometry f_{iso} to transform $\mathcal{D}_1$ and integrate it with $\mathcal{D}_2$. (c) Hypothesis B acknowledges that vector databases do not have global geometry consistency for $\mathcal{O}_\cap$; it instead suggests that we exploit multiple local isometries between $\mathcal{D}_1[\mathcal{O}_\cap]$ with $\mathcal{D}_2[\mathcal{O}_\cap]$, enabling integration even if Hypothesis A does not hold.

revised local isometry hypothesis in Section 4.1, based on which we develop our vector database integration framework (Section 4.2). We prove an upper bound on the integration error of the framework (Section 5), which guides its implementation (Section 6). We experimentally verify the effectiveness of our approach with 5 public datasets and 6 major embedding models (Section 7). We discuss additional related research in Section 8 and conclude in Section 9.

2 Integrating Vector Databases

Vector databases. Denote by $\mathcal{U}$ the universe of data items, of which the *dissimilarity* between items is measured by function $\texttt{dsim}$. Let $\mathcal{O}$ be a subset of $\mathcal{U}$. An embedding function $\texttt{emb}$ for $\mathcal{O}$ *w.r.t.* $\texttt{dsim}$ maps items in $\mathcal{O}$ to high-dimensional vectors $\mathbb{R}^d$ such that $\|\mathbf{x}_1 - \mathbf{x}_2\|$ encodes $\texttt{dsim}(o_1, o_2)$ for any $o_1, o_2 \in \mathcal{O}$ with $\texttt{emb}(o_1) = \mathbf{x}_1$ and $\texttt{emb}(o_2) = \mathbf{x}_2$, where $\|\mathbf{x}_1 - \mathbf{x}_2\|$ denotes the distance between $\mathbf{x}_1$ and $\mathbf{x}_2$ in $\mathbb{R}^d$. We refer to $\texttt{emb}(\mathcal{O})$, *i.e.,* $\cup_{o\in\mathcal{O}}\texttt{emb}(o)$, as a *vector database of* $\mathcal{O}$ *with* $\texttt{emb}$ *w.r.t.* $\texttt{dsim}$.

The goal of vector databases is to implement top-k similarity search: given item $o \in \mathcal{O}$, find k most similar items in $\mathcal{O}$ *w.r.t.* $\texttt{dsim}$, by searching for k vectors in $\texttt{emb}(\mathcal{O})$ that are nearest to $\texttt{emb}(o)$.

In practice, vector distances in $\texttt{emb}(\mathcal{O})$ are not exactly equivalent to dissimilarity of embedded items. Hence, $\|\mathbf{x}_1 - \mathbf{x}_2\|$ is only an approximation of $\texttt{dsim}(o_1, o_2)$ as long as top-k vector search over $\texttt{emb}(\mathcal{O})$ implements top-k similarity search sufficiently accurate.

Vector database integration. Let $\mathcal{O}_1$ and $\mathcal{O}_2$ be subsets of $\mathcal{U}$, *i.e.,* two different datasets. Denote by $\mathcal{D}_1$ (resp. $\mathcal{D}_2$) the vector database of $\mathcal{O}_1$ (resp. $\mathcal{O}_2$) with embedding function $\texttt{emb}_1$ (resp. $\texttt{emb}_2$) *w.r.t.* $\texttt{dsim}$. The *integration of* $\mathcal{D}_1$ *and* $\mathcal{D}_2$ is a function $\mathcal{T}$ that takes as input only vectors of $\mathcal{D}_1$ and $\mathcal{D}_2$, and returns an vector database $\mathcal{T}(\mathcal{D}_1, \mathcal{D}_2)$ of $\mathcal{O}_1 \cup \mathcal{O}_2$ with some embedding $\texttt{emb}_\mathcal{T}$ *w.r.t.* $\texttt{dsim}$.

Intuitively, $\mathcal{T}$ seeks to combine separate vector databases $\mathcal{D}_1$ and $\mathcal{D}_2$ into a merged database $\mathcal{D}_\cup$ that encodes $\texttt{dsim}$ of data across $\mathcal{O}_1 \cup \mathcal{O}_2$ via some embedding function $\texttt{emb}_\mathcal{T}$. This implies that, for any items $o, o' \in \mathcal{O}_1 \cup \mathcal{O}_2$, vector distance $\|\texttt{emb}_\mathcal{T}(o) - \texttt{emb}_\mathcal{T}(o')\|$ in $\mathcal{D}_\cup$ is, approximately,

- $\|\texttt{emb}_1(o) - \texttt{emb}_1(o')\|$ if both o and o' are in $\mathcal{O}_1$ (agrees with $\mathcal{D}_1$);
- $\|\texttt{emb}_2(o) - \texttt{emb}_2(o')\|$ if both o and o' are in $\mathcal{O}_2$ (agrees with $\mathcal{D}_2$);
- $\texttt{dsim}(o, o')$ if o and o' are from different datasets.

If this is doable, it would allow us to construct a *global* vector database for a federation of disparate datasets without accessing their raw data records. We can then expand top-k similarity search from a single dataset to the whole federation, via top-k vector search over the integrated vector database.

While this is desirable, it is however challenging to find such a function $\mathcal{T}$ for a few reasons.

(a) $\mathcal{T}$ must combine $\mathcal{D}_1$ and $\mathcal{D}_2$ into a vector database for $\mathcal{O}_1 \cup \mathcal{O}_2$ *w.r.t.* $\texttt{dsim}$ by operating on $\mathcal{D}_1$ and $\mathcal{D}_2$ only, *without* accessing $\mathcal{O}_1$ or $\mathcal{O}_2$, nor should it be aware of their respective embedding functions.

(b) It has to capture both $\texttt{emb}_1$ and $\texttt{emb}_2$ by agreeing on their vector distances for $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively, rather than only fitting $\mathcal{D}_1$ ($\texttt{emb}_1$) into $\mathcal{D}_2$ ($\texttt{emb}_2$). This is crucial as we need to *retain* accurate top-k similarity search over items only in $\mathcal{O}_1$ or only in $\mathcal{O}_2$, and *generate* dissimilarity between items across $\mathcal{O}_1$ and $\mathcal{O}_2$ that has not been encoded in either $\mathcal{D}_1$ or $\mathcal{D}_2$.

(c) Top-k search is particularly sensitive to *short distances* as it typically involves finding a limited number (k) of results that are close to the query vector. Consequently, even minor variations in these short distances can significantly impact the search results, even when the global distribution of the dataset remains unchanged. This indicates that $\mathcal{T}$ must preserve short distances in $\mathcal{D}_1$ and $\mathcal{D}_2$.

(d) Most critically, the integration of $\mathcal{D}_1$ and $\mathcal{D}_2$ is not always guaranteed. This assumes that both $\mathcal{D}_1$ and $\mathcal{D}_2$ encode the same $\texttt{dsim}$ over their respective datasets $\mathcal{O}_1$ and $\mathcal{O}_2$. Even when they do, $\mathcal{O}_1$ and $\mathcal{O}_2$ have to be associable for $\mathcal{T}$ to exist. We cannot infer or generate the correct dissimilarity between data items across $\mathcal{O}_1$ and $\mathcal{O}_2$ if they have no overlapping items, as $\mathcal{T}$ is only exposed to the encoded dissimilarity between items both within $\mathcal{O}_1$ or both within $\mathcal{O}_2$. When $\mathcal{D}_1$ and $\mathcal{D}_2$ are disjoint or overlap too little, there is too weak an association between them to determine if $\mathcal{D}_1$ and $\mathcal{D}_2$ observe the same dissimilarity function $\texttt{dsim}$, thus not able to establish function $\mathcal{T}$ to integrate $\mathcal{D}_1$ and $\mathcal{D}_2$.

Remark. In practice, vector distances are only approximations of the dissimilarity between embedded data items. Thus, whether $\mathcal{D}_1$ and $\mathcal{D}_2$ are valid databases or if their encoded dissimilarity $\texttt{dsim}_1$ and $\texttt{dsim}_2$ agree over $\mathcal{O}_1 \cup \mathcal{O}_2$ depends on our tolerance for the inaccuracies of vector distances, which ultimately is reflected in the accuracy of approximating top-k similarity search. To simplify presentation and focus on the main technical challenge of finding $\mathcal{T}$, we will always assume that $\mathcal{D}_1$ and $\mathcal{D}_2$ can be integrated, and we will reflect the different extents of mergeability by their top-k similarity search accuracy over benchmarks with curated ground truth.

Hypothesis. From the discussion above, we know that for $\mathcal{D}_1$ and $\mathcal{D}_2$ to be mergeable, $\mathcal{O}_1$ and $\mathcal{O}_2$ must share sufficiently many data items, *i.e.,* $\mathcal{O}_\cap = \mathcal{O}_1 \cap \mathcal{O}_2$ is sufficiently large, and $\mathcal{D}_1$ and $\mathcal{D}_2$ shall agree on dissimilarity over $\mathcal{O}_\cap$. This seems to suggest that $\mathcal{D}_1[\mathcal{O}_\cap]$ (vectors of $\mathcal{D}_1$ that encode $\mathcal{O}_\cap$) and $\mathcal{D}_2[\mathcal{O}_\cap]$ are approximately *isometric* [31]: there exists a distance-preserving bijective mapping f_{iso} between $\mathcal{D}_1[\mathcal{O}_\cap]$ and $\mathcal{D}_2[\mathcal{O}_\cap]$. This motivates us to consider the following hypothesis.

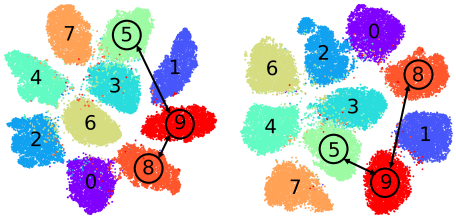
Hypothesis A [The Isometry Hypothesis]: *(1) The geometry shapes of vector databases for the same set $\mathcal{O}$ of data items embedded by different functions $\texttt{emb}_1$ and $\texttt{emb}_2$ are likely isometric.*

(2) If $\texttt{emb}_1$ and $\texttt{emb}_2$ embed $\mathcal{O}$ into isometric vector databases, they will also likely embed data items close to those in $\mathcal{O}$ into vectors that have approximately isometric geometry shapes. □

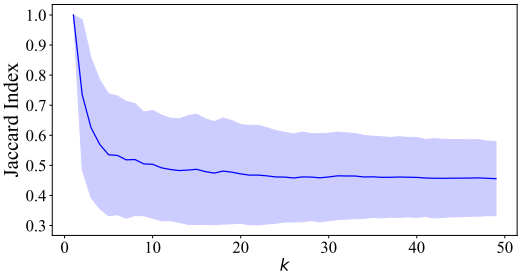
If Hypothesis A(1) holds, we then know that $\mathcal{D}_1$ and $\mathcal{D}_2$ embedded by $\texttt{emb}_1$ and $\texttt{emb}_2$, respectively, agree on the dissimilarity over $\mathcal{O}_\cap$. With sufficiently large $\mathcal{O}_\cap$, this implies that $\mathcal{D}_1$ and $\mathcal{D}_2$ encode the same $\texttt{dsim}$ over their respective datasets $\mathcal{O}_1$ and $\mathcal{O}_2$. When Hypothesis A(2) also holds, this means that $\texttt{emb}_2$ would agree with vector distances in $\mathcal{D}_1$ if it were used to embed $\mathcal{O}_1 \setminus \mathcal{O}_2$.

Align-to-merge. This is particularly helpful as it suggests an "*align-to-merge*" ($\texttt{A2M}$) function $\mathcal{T}$ to integrating $\mathcal{D}_1$ and $\mathcal{D}_2$: we first compute the isometry f_{iso} that maps $\mathcal{D}_1[\mathcal{O}_\cap]$ to $\mathcal{D}_2[\mathcal{O}_\cap]$, and then use f_{iso} to transform and merge $\mathcal{D}_1$ into $\mathcal{D}_2$, yielding the integrated vector database as $f_{\text{iso}}(\mathcal{D}_1) \cup \mathcal{D}_2[\bar{\mathcal{O}}_\cap])$, where $\mathcal{D}_2[\bar{\mathcal{O}}_\cap]$ are embedding vectors of $\mathcal{D}_2$ that do not correspond to items in $\mathcal{O}_\cap$.

This gives us an integration function $\mathcal{T}$ that only transforms $\mathcal{D}_1$ while keeping $\mathcal{D}_2$ intact, *i.e.,* $\texttt{emb}_\mathcal{T}$ is exactly $\texttt{emb}_2$. Thus one can then use vectors embedded by $\texttt{emb}_2$ as queries to search the



(a) T-SNE visualization of CIFAR-10 **[47] with embedding models VGGNet [72] (left) and DenseNet [40] (right):** hand-written '5' images are embedded further away from '9' by VGGNet while the opposite is observed in DenseNet embeddings.



(b) Consistency of top-k search over SciFact **with NV-Embed V2 and** `OpenAI-Ada` (blue line is the average of 10 random queries; shaded area shows the variance.)

Fig. 3. Are real-life vector databases (approximately) isometric?

integrated database. Moreover, the isometry f_{iso} is easy to deduce by aligning $\mathcal{D}_1[O_\cap]$ with $\mathcal{D}_2[O_\cap]$ via embedding alignment, a core task in robotics (rigid motion transformation) [3], computer vision (Procrustes Analysis) [69, 76] and cross-lingual word embedding alignment [5, 44, 63, 74]. We illustrate such integration function $\mathcal{T}$ by drawing parallel with image object correlation in Fig. 2 (a-b).

3 Evaluating the Isometry Hypothesis

While we have a viable solution for vector database integration by making a connection with embedding alignment via isometries, it critically depends on the assumption that Hypothesis A holds for vector databases, which we will examine closely next. Specifically, we aim to answer two questions (Q1) and (Q2) below, corresponding to part (1) and part (2) of Hypothesis A, respectively:

(Q1) Do different embedding models generate (approximately) isometric vectors of the same data?

(Q2) How well does top-k vector search perform over the integrated vector databases based on the align-to-merge approach suggested by Hypothesis A (as also illustrated in Fig. 2b)?

3.1 Are Vector Databases Isometric (Q1)?

We first examine question (Q1) to see if the same dataset (*i.e.,* $O_\cap$ in Section 2) has approximately isometric vector representations embedded by different models. We use two evaluation methods: (a) we visualize vector databases of the same dataset with T-SNE [78] and examine if their visualizations are isometric; and (b) we compare the top-k similarity search results implemented by different vector databases of the same dataset. We used both image and text data.

(1) Visualizing vector databases. We first visualize two vector databases embedded from the CIFAR-10 dataset, which is commonly used for evaluating image classification methods [47].

More specifically, CIFAR-10 consists of consists of 60,000 color images categorized into 10 distinct classes. We use all the images as the set O of data items, and generate two vector databases $\texttt{emb}_1(O)$ and $\texttt{emb}_2(O)$ of O with ResNet [37] and DenseNet [39], respectively. The T-SNE [78] visualization of $\texttt{emb}_1(O)$ and $\texttt{emb}_2(O)$ is shown in the Fig. 3a. Vectors of images from the same category are highlighted in the same color in both vector databases, and are numbered from 0 to 9. We examine the proximity between the classes in the T-SNE visualization of both vector databases such that small noises in the embedding vectors are subsumed within the classes.

Observation. If Hypothesis A holds, we would see that the classes are distributed similarly in the visualization across the two vector databases. However, this is not the case as shown by Fig. 3a. For

instance, class 8 is closer to class 9 than class 5 in VGGNet embedding vectors, while the opposite is shown in DenseNet. This shows that, VGGNet and DenseNet vectors are not isometric even if we ignore perturbation in short vector distances and zoom out to the class level to subsume distance noise.

(2) Consistency of top-k similarity search. We further examine the isometry hypothesis by looking into the consistency of top-k similarity search. The rationale is simple: if two vector databases $\texttt{emb}_1(O)$ and $\texttt{emb}_2(O)$ of the same dataset O produced by embedding models $\texttt{emb}_1$ and $\texttt{emb}_2$, respectively, are isometric, for any query item $q \in O$, its top-k similarity search, implemented as top-k vector search of $\texttt{emb}_1(q)$ over $\texttt{emb}_1(O)$ and that of $\texttt{emb}_2(q)$ over $\texttt{emb}_2(O)$ shall be consistent and identify the same objects that are deemed as the most similar to q in O, where $\texttt{emb}_1(q)$ and $\texttt{emb}_2(q)$ are the vectors of o embedded by $\texttt{emb}_1$ and $\texttt{emb}_2$, respectively.

To do this, we use the SciFact text dataset [80] as O, which consists of 6,000 scientific article abstracts. We use embedding models `NV-Embed-V2` [50] and `OpenAI-Ada` [66] as $\texttt{emb}_1$ and $\texttt{emb}_2$ to create two vector databases $\texttt{emb}_1(O)$ and $\texttt{emb}_2(O)$ for SciFact, respectively. To measure the consistency of top-k search results, we use Jaccard Index [17]. Specifically, for an item $q \in O$, let $R_1 \subseteq O$ be the k items that are encoded by the top-k vector search results with query vector $\texttt{emb}_1(q)$ over $\texttt{emb}_1(O)$; similarly, let $R_2 \subseteq O$ be the items identified by $\texttt{emb}_2(q)$ over $\texttt{emb}_2(O)$. Then we define the *consistency* of top-k search with $\texttt{emb}_1(O)$ and $\texttt{emb}_2(O)$ for q as the Jaccard Index of sets R_1 and R_2, calculated as $|R_1 \cap R_2|/|R_1 \cup R_2|$.

Intuitively, the consistency of top-k search with $\texttt{emb}_1(O)$ and $\texttt{emb}_2(O)$ ranges in $[0, 1]$ and a larger consistency ratio indicates a higher degree of alignment between the retrieval outcomes from the two different embedding representations of O.

Observation. Figure 3b shows the average consistency ratio between top-k results over `NV-Embed-V2` vectors and Openai Ada vectors for 10 random query questions, with varying k. The results tell us that Hypothesis A does not hold for `NV-Embed-V2` and `OpenAI-Ada` embedding vectors, as the consistency between search results over `NV-Embed-V2` and `OpenAI-Ada` embedding vectors is mostly low (around 0.5), unless k is very small (e.g., less than 5).

Answer to Q1. From these experiments, we observe that vector databases of the same set of data items are not necessarily isometric if they are generated by different embedding models. Moreover, from (1) we can see that this non-isometric phenomenon cannot be simply attributed to embedding noises that purturbate the "positions" of the vectors in the space as they are not isometric even if we zoom out and view the dataset from a quite coarse resolution.

3.2 Vector Search over Integrated Databases (Q2)

We next examine evaluation question (Q2) to see how well top-k vector search performs over the integrated vector database produced by align-to-merge approach sketched in Section 2. We first present the approach in detail and then evaluate its performance for Q2.

Align-to-merge (`A2M`) for vector database integration. Recall notations from Section 2 that O_1 and O_2 are two sets of objects, encoded by two embedding functions $\texttt{emb}_1$ and $\texttt{emb}_2$ as vector databases $\mathcal{D}_1$ and $\mathcal{D}_2$, respectively. Let $O_\cap = O_1 \cap O_2$. Denote by $\mathcal{X}$ and $\mathcal{Y}$ the vectors in $\mathcal{D}_1$ and $\mathcal{D}_2$ that encode objects in $O_\cap$, respectively. Assume *w.l.o.g.* that $\mathcal{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ and $\mathcal{Y} = \{\mathbf{y}_1, \ldots, \mathbf{y}_m\}$, and vectors $\mathbf{x}_i$ and $\mathbf{y}_i$ encode the same object in $O_\cap$ ($i \in [1, m]$).

The align-to-merge (`A2M`) approach operates in two steps. In the first step, it computes a best fit isometry function f_{iso} from $\mathcal{X}$ to $\mathcal{Y}$, as shown in Fig. 2b, to align $\mathcal{D}_1$ with $\mathcal{D}_2$ by using the fact that their subsets $\mathcal{X}$ and $\mathcal{Y}$ encode the same objects in $O_\cap$. According to Hypothesis A, when $O_\cap$ is sufficiently large, it gives us a mapping f_{iso} that determines an integration of $\mathcal{D}_1$ into $\mathcal{D}_2$ as $f_{\text{iso}}(\mathcal{D}_1) \cup (\mathcal{D}_2 \setminus \mathcal{Y})$, serving as an approximation of the vector database that $\texttt{emb}_2$ would generate for $O_\cap$.

Following Procrustes Analysis in computer vision [69, 76] and cross-lingual word embedding alignment [44, 74], f_{iso} can be formulated as $f(\mathbf{x}) = s*R(\mathbf{x}+\mathbf{t})$, where R is an orthogonal matrix of dimension $d\times d$ that represents rotation and reflection, $\mathbf{t}$ is a d-dimensional vector for translation, and s is for uniform scaling, and finding the best f_{iso} that maps $\mathcal{X}$ to $\mathcal{Y}$ reduces to an optimization problem:

$$\arg\min_{s,R,\mathbf{t}} \textstyle\sum_{i=1}^{m} \|sR(\mathbf{x}_i + \mathbf{t}) - \mathbf{y}_i\|^2, \tag{1}$$

where $\|.\|$ is the L2 norm of vectors. Intuitively, f_{iso} transforms $\mathcal{X}$ via isometry (*i.e.,* R and $\mathbf{t}$) extended with uniform scaling (*i.e.,* s) such that vectors of $\mathcal{X}$ are best aligned with their targets in $\mathcal{Y}$, with the sum of square of mis-alignment distances minimized. When $\mathcal{X}$ is exactly isometric to $\mathcal{Y}$, $f_{\text{iso}}(\mathbf{x}_i)$ is $\mathbf{y}_i$.

A nice property of this formulation is that it has a closed-form optimal solution. Let the SVD (Singular Value Decomposition) [24] of $\mathcal{Y}\mathcal{X}^T$ be $U\Sigma V^T$, where we view $\mathcal{X}$ as a $d \times m$ matrix with $\mathbf{x}_i (i \in [1, m])$ as column vectors; similarly, $\mathcal{Y}$ is a $d \times m$ matrix that has $\mathbf{y}_1, \ldots, \mathbf{y}_m$ as its column vectors. Then f_{iso} is determined by the optimal R, $\mathbf{t}$ and s as $f_{\text{iso}}(\mathbf{x}) = s_{\text{opt}} * R_{\text{opt}}(\mathbf{x} + \mathbf{t}_{\text{opt}})$:

- $R_{\text{opt}} = UV^T$, where $\text{SVD}(\mathcal{Y}\mathcal{X}^T) = U\Sigma V^T$,
- $s_{\text{opt}} = \text{trace}(\Sigma)/\text{trace}(\mathcal{X}^T\mathcal{X})$, and
- $\mathbf{t}_{\text{opt}} = \bar{\mathbf{y}} - \bar{\mathbf{x}}$, where $\bar{\mathbf{x}} = \sum_{i=1}^{m} \mathbf{x}_i/m$ and $\bar{\mathbf{y}} = \sum_{i=1}^{m} \mathbf{y}_i/m$.

Evaluation of Q2. We next examine evaluation question (Q2) to see how well top-k vector search performs over the integrated vector database $f_{\text{iso}}(\mathcal{D}_1) \cup (\mathcal{D}_2 \setminus \mathcal{Y})$ computed by A2M.

To do this, we use four public text retrieval benchmarks [77], each consisting of a set of text documents along with built-in query-answer pairs (q, ans), where q is a query text and ans is the ground-truth answer documents to q validated by the benchmark.

We randomly partition each dataset into $\mathcal{O}_1$ and $\mathcal{O}_2$ with $\mathcal{O}_1 \cap \mathcal{O}_2$ containing randomly picked documents accounting for 1/3 of the dataset size; built-in benchmark ans documents are evenly distributed in $\mathcal{O}_1$ and $\mathcal{O}_2$. We then use two embedding models $\texttt{emb}_1$ and $\texttt{emb}_2$ to encode $\mathcal{O}_1$ and $\mathcal{O}_2$, respectively, as $\mathcal{D}_1$ and $\mathcal{D}_2$. Denote by $\mathcal{D}_{12}$ the database computed by A2M, by integrating $\mathcal{D}_1$ into $\mathcal{D}_2$. We evaluated the quality of $\mathcal{D}_{12}$ by the recall@k of top-k vector search with query vector $\texttt{emb}_2(q)$, *i.e.,* the vector of q embedded by $\texttt{emb}_2$. Here recall@k is the percentage of (q, ans) pairs with ans documents identified by the top-k vector search with $\texttt{emb}_2(q)$ over D_{12}.

The results of for various $\texttt{emb}_1$ and $\texttt{emb}_2$ models are shown in Table 1 (without the part shadowed in gray; see more in Section 7). Here Union is a baseline that directly takes the union of $\mathcal{D}_1$ and $\mathcal{D}_2$: *i.e.,* it uses identify function as f_{iso} in the A2M framework.

The results show that text retrieval quality over vectors integrated by A2M is similar or even worse (*e.g.,* on SciFact benchmark [80]) than that of Union. Indeed, Union guarantees that all ans documents that are distributed only in $\mathcal{O}_2$ are recalled via $\texttt{emb}_2(q)$ over the integrated database, since all vectors of $\mathcal{D}_1$ would be disregarded during retrieval as they are far away from vectors of $\mathcal{D}_2$. On the other hand, A2M transforms and places vectors of $\mathcal{D}_1$ in close proximity to vectors in $\mathcal{D}_2$, which can affect the recall performance over integrated database if they are not perfectly placed.

Answer to Q2. We observe that integrating $\mathcal{D}_1$ into $\mathcal{D}_2$ based on the Hypothesis A (via A2M) does not help improve the recall of vector search over $\mathcal{D}_2$ even though integration expands similarity search across datasets. As we will see in Section 7, the results are similar even if we replace the simple isometry f_{iso} with more sophisticated or even non-linear learned embedding alignment functions. This is consistent with the evaluation result of (Q1) that suggests different embedding vectors of the same objects $\mathcal{O}_\cap$ are not really isometric, rendering $f_{\text{iso}}(\mathcal{D}_1)$ inaccurate for $\mathcal{D}_2$.

Table 1. Recall@100 over integrated vector databases

$\texttt{emb}_1$ **vs** $\texttt{emb}_2$	**method**	**Dataset**			
		SciFact	NFCorpus	ArguAna	SciDocs
Fast Text **vs** NV-embed V2	Union	26.33	13.62	30.34	10.80
	A2M	18.33	8.05	25.91	5.70
	LA2M	76.67	34.06	64.60	39.30
NV-embed **vs** Openai Ada	Union	32.00	14.24	28.19	16.70
	A2M	32.00	14.24	28.19	16.70
	LA2M	79.33	35.29	85.22	41.10
gte-Qwen2 **vs** NV-embed V2	Union	31.67	18.27	28.48	19.60
	A2M	32.33	15.48	28.48	18.40
	LA2M	85.33	31.27	86.65	44.60
Mistral **vs** Openai Ada	Union	32.00	15.17	28.19	17.60
	A2M	27.00	10.84	29.91	12.60
	LA2M	76.67	34.06	79.23	42.00

4 Hypothesis Revisited for Vector Database Integration

From Section 3, we observe that Hypothesis A does not necessarily hold for vector databases. As a result, one cannot simply reduce the integration of vector databases to an application of existing embedding alignment techniques via A2M as this would yield an integrated database that cannot accurately implement top-k similarity search.

To rectify this, we first re-examine the hypothesis, showing that all we need to make Hypothesis A work for vector databases is a simple condition on vector distances of concern (Section 4.1). Based on it we then develop a new integration framework (Section 4.2).

4.1 Hypothesis: From Global Isometry to Local Isometries

We have seen from Section 3.2 that the integration of vector databases $\mathcal{D}_1$ and $\mathcal{D}_2$ for data items of O_1 and O_2, respectively, cannot serve as a vector database that encodes the universe $O_1 \cup O_2$, as shown by the performance of A2M in Table 1. This is mostly due to that different embedding models produce non-isometric vectors for common data items in $O_1 \cap O_2$ as shown in Fig. 3a of Section 3.1.

However, we are not completely fruitless. In fact, the experimental findings, especially those in Fig. 3b, seem to suggest that different embedding models behave quite consistently for *short distances*. Such short distances are also the key to implement accurate top-k similarity search for reasonable k.

As shown in Fig. 3b, over SciFact dataset when k is small, *e.g.,* less than 5, top-k document retrieval results with NV-Embed-V2 embeddings and those with OpenAI-Ada are almost identical, with Jaccard Index consistently approaching 1. When k increases, this consistency between NV-Embed-V2 and OpenAI-Adadecreases significantly. This seems to suggest that NV-Embed-V2 and OpenAI-Ada encode closely related documents via consistent short vector distances, despite that they are entirely different embedding models. We capture this observation via a new hypothesis below.

Hypothesis B [The Local Isometry Hypothesis]: *The global isometry hypothesis (Hypothesis A) holds within local regions of vector databases comprised of vectors with short distances.* □

To validate if this holds, we first visually examine local structures of vector databases embedded by FastText [8] and NV-Embed-V2 [50] for the SciFact dataset. The zoomed-in T-SNE visualization of a query and its 10 nearest neighbors is shown in Fig. 4, where the left side depicts the distribution of results identified by top-10 vector search over FastText embedding vectors, and the right side shows the results of NV-Embed-V2 vectors. It is not difficult to see that the distribution of the top-k vector search results of the two embedding models are strikingly similarly: they produce exactly

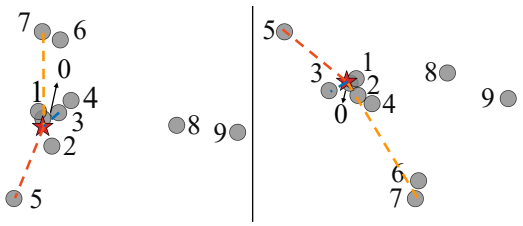


Fig. 4. A zoomed-in T-SNE view of a query (★) and its 10 nearest neighbors in SciFact **(**left shows the results over `FastText` [91] embeddings and right depicts `NV-Embed-V2` [50] results.)

the same list of top-10 most similar SciFact documents, and these answers form an approximately isometric shape in the vector space. This seems to suggest that Hypothesis A(1) indeed holds in this local region of SciFact vectors embedded by `FastText` and `NV-Embed-V2`

To further evaluate if Hypothesis A(2) holds in local regions as Hypothesis B states above, we note that Hypothesis B naturally suggests a new approach to integrating vector databases produced by different embedding models, which we refer to as *localized* `A2M` (denoted by `LA2M` and illustrated in Fig. 2c): (1) we first decompose vector databases into clusters of vectors that have short inter-distances; (2) we then apply `A2M` of Section 3.2 independently to each cluster.

We then test the accuracy of integrated vector databases computed by `LA2M`, by using exactly the same setting as we evaluate `A2M` in Section 3.2. The results, as highlighted in gray in Table 1, show that `LA2M` significantly boosts the accuracy of `A2M`. For instance, its recall@100 of document retrieval results over integrated vector databases is consistently more than double of that of `A2M` and `Union` (see more results in Section 7). This further confirms that Hypothesis A(2) indeed holds in local regions of vector databases. These findings indicate that, for real-life vector databases, Hypothesis B is significantly more valid than Hypothesis A.

4.2 A Framework for Vector Database Integration

We next describe framework `LA2M` in detail, showing how we make "localized" use of embedding alignment to best exploit Hypothesis B for better integrated databases. We also point out key technical challenges in using `LA2M`, which we will tackle in subsequent sections.

Framework. `LA2M` takes as input two vector databases $\mathcal{D}_1$ and $\mathcal{D}_2$ and produces an integration mapping function $\mathcal{T}$ such that $\mathcal{T}(\mathcal{D}_1) \cup \mathcal{D}_2$ serves as an integrated vector database for data items encoded by $\mathcal{D}_1$ and $\mathcal{D}_2$ taken together. Here $\mathcal{D}_1$ and $\mathcal{D}_2$ are possibly generated by two different embedding models $\texttt{emb}_1$ and $\texttt{emb}_2$, respectively, for data item collections $\mathcal{O}_1$ and $\mathcal{O}_2$; however, `LA2M` assumes no access or knowledge to either $\texttt{emb}_i$ or $\mathcal{O}_i (i \in \{1, 2\})$ when computing the integration function $\mathcal{T}$.

It does so by exploiting *reference* vectors between $\mathcal{D}_1$ and $\mathcal{D}_2$: pairs of vectors that encode the same objects, *i.e.,* $\mathcal{X}$ and $\mathcal{Y}$ in `A2M` of Section 3.2. Reference vectors are abundant in scenarios where we need to integrate vector databases, as databases must be relevant so that integration is beneficial. Typical sources of references include known correspondence in embedding backfilling, domain knowledge, or data linkage results. In case they are not know as a priori, `LA2M` allows to automatically deduce such references (Section 6).

More specifically, `LA2M` integrates $\mathcal{D}_1$ into $\mathcal{D}_2$ in three steps:

Step (1): *Grouping*. It first groups the reference vectors $\mathcal{X}$ in $\mathcal{D}_1$ into clusters $X_1, \ldots, X_n$, each with vectors that have short inter-distances. Denote by Y_i the set of matching vectors of X_i in $\mathcal{Y}$ of $\mathcal{D}_2$ ($i \in [1, n]$). Note that the clusters do not have to be disjoint.

Step (2): *Alignment.* Following Hypothesis B, LA2M deduces a collection of isometries to align the reference clusters independently. For each pair X_i and Y_i, it invokes A2M to compute an isometry f^i_{iso} that aligns X_i to Y_i as discussed in Section 3.2. One may also replace A2M with other embedding alignment variants; we use isometries here as they are easy to compute and offer us a provable bound on the quality of integrated vector database as we will see in Section 5.

Step (3): *Merging.* LA2M decides the integration mapping $\mathcal{T}$ for $\mathcal{D}_1$ with local isometries computed in step (2). Specifically, for any vector $\mathbf{x}$ in $\mathcal{D}_1$, $\mathcal{T}(\mathbf{x})$ maps $\mathbf{x}$ according to the position of $\mathbf{x}$:

(a) $\mathbf{x}$ *is a reference vector.* When $\mathbf{x}$ is a reference vector in a cluster X_i grouped in step (1), $\mathcal{T}$ maps $\mathbf{x}$ to the vector $\mathbf{y}$ in Y_i of $\mathcal{D}_2$ that encodes the same data item as $\mathbf{x}$ does in X_i of $\mathcal{D}_1$.

(b) $\mathbf{x}$ *is a non-reference vector.* Otherwise, $\mathbf{x}$ is a non-reference vector in $\mathcal{D}_1$ that needs to be mapped to the vector space of $\mathcal{D}_2$. $\mathcal{T}$ does so by picking X_{j^*} with centroid $\mathbf{c}_{j^*}$ that is closest to $\mathbf{x}$ among all reference clusters, *i.e.,* $j^* = \arg\min_{i\in[1,n]} \|\mathbf{c}_i - \mathbf{x}\|$, where $\mathbf{c}_{j^*} = \sum_{\mathbf{x}\in X_{j^*}} \mathbf{x}/|X_{j^*}|$. It then uses $f^{j^*}_{\text{iso}}$ to map $\mathbf{x}$ to $\mathcal{D}_2$, *i.e.,* $\mathcal{T}(\mathbf{x}) = f^j_{\text{iso}}(\mathbf{x})$.

Finally, LA2M returns $\mathcal{T}(\mathcal{D}_1) \cup (\mathcal{D}_2 \setminus \mathcal{Y})$ as the integrated database. With access to $\mathcal{D}_1$ and $\mathcal{D}_2$ only, LA2M expands the scope of top-k similarity search from over only data items encoded by $\mathcal{D}_2$ to over the universe of objects encoded by $\mathcal{D}_1$ and $\mathcal{D}_2$ taken together.

Benefits & implications. As shown in Table 1, LA2M enables us to integrate vector databases across embedding models, realizing cross-dataset similarity search. It also has diverse implications.

(1) Model & object oblivious. LA2M merges vector databases without integrating the data items encoded by the vectors, nor requiring access to the embedding models. Such obliviousness to the actual encoded datasets and embedding models makes LA2M a perfect fit for cases where we cannot freely share objects but still want to search across datasets, *e.g.,* when sharing data objects can be expensive such as in edge devices or restricted due to privacy constraints.

(2) Vector index invariant. LA2M not only transforms and maps vectors of $\mathcal{D}_1$ to $\mathcal{D}_2$, it can also directly transforms vector indices built atop $\mathcal{D}_1$, *e.g.,* IVF and HNSW, to the transformed vectors in $\mathcal{D}_2$ space. This is because LA2M uses isometries to align and merge vectors. In addition to preserving similarity search over $\mathcal{D}_1$, isometries can also transform IVF and HNSW over $\mathcal{D}_1$ as they are based on vector distances which are preserved during the merging step.

Key factors & challenges. The quality of the integrated database $\mathcal{T}(\mathcal{D}_1) \cup \mathcal{D}_2$ heavily relies on the following two factors of LA2M:

- *Availability of references* $\mathcal{X} \subseteq \mathcal{D}_1$ and $\mathcal{Y} \subseteq \mathcal{D}_2$.
- *Scope of isometries*: grouping $\mathcal{X}$ in to $X_1, \ldots, X_n$.

LA2M decides the isometries by analyzing reference vectors $\mathcal{X}$ and $\mathcal{Y}$ across $\mathcal{D}_1$ and $\mathcal{D}_2$. Such references are available from domain knowledge on the datasets, or can be deduced by employing entity linking methods [10, 71]. The rationale is that when we want to integrate $\mathcal{D}_1$ and $\mathcal{D}_2$, the data items encoded by them must be correlated; such correlation offers potential for the existence of linkage across the datasets, which, mapped to the vectors, gives us the reference vectors $\mathcal{X}$ and $\mathcal{Y}$. In the worst case where $\mathcal{X}$ and $\mathcal{Y}$ are not available, we develop an interface for LA2M to actively generate references with minimum access to the datasets (Section 6.2).

Even more challenging is to decide how we shall group $\mathcal{X}$ into local clusters $X_1, \ldots, X_n$, to maximize the quality of integrated vector database $\mathcal{T}(\mathcal{D}_1) \cup \mathcal{D}_2$. As shown in Fig. 5, the scope of X_i evidently affects the errors that LA2M produces in step (2) and step (3), which we refer to as the *alignment* error α and *integration* error β, respectively. To deal with this, we first theoretically

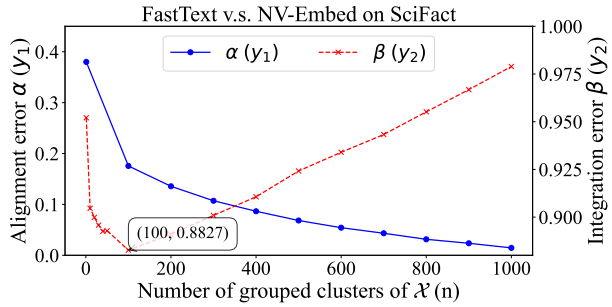


Fig. 5. #-clusters of $\mathcal{X}$ vs alignment error α and integration error β. (alignment error α: distance between $f_{\text{iso}}^{j}(\mathbf{x})$ and $\mathbf{y}$ when $\mathbf{x} \in X_j$ corresponds to $\mathbf{y} \in Y_j$ (step (2) error); β: distance between $\mathcal{T}(\mathbf{x})$ and the vector would have been embedded by $\texttt{emb}_2$ (step (3) error). $\mathcal{X}$ is grouped with k-meaning clustering.)

analyze and bound the errors on mappings $\mathcal{T}(\mathbf{x})$ of vectors $\mathbf{x} \in \mathcal{D}_1$ in Section 5, based on which we then discuss how LA2M breaks down $\mathcal{X}$ in Section 6.1.

5 Bounding the Quality of Integrated Vector Databases

To decide key design factors of LA2M, we study the quality of integrated vector databases.

Quantifying integration quality. We first consider metrics to quantify the quality of vector databases integrated by LA2M. While recall@k as reported in Table 1 (and also Section 7) is a standard choice for similarity search, it is sensitive to the curated query-answer samples built in the benchmarks; moreover, its empirical nature offers little insight in the source and properties of the errors.

To this end, below we use vector distances directly to quantify the error of every transformed vector of $\mathcal{D}_1$ in the vector space of $\mathcal{D}_2$. Recall from Section 4.2 that LA2M transforms vectors of $\mathcal{D}_1$ embedded by model $\texttt{emb}_1$ and merges them into database $\mathcal{D}_2$ embedded by model $\texttt{emb}_2$ via function $\mathcal{T}$, in the form of a collection of isometries. Each isometry f_{iso}^{i} is determined by a local cluster X_i of reference vectors that corresponds to its counterpart Y_i in $\mathcal{D}_2$.

Integration error (β). Consider an arbitrary non-reference vector $\mathbf{x}$ of $\mathcal{D}_1$. We call $\mathcal{T}(\mathbf{x})$ its image *w.r.t.* $\mathcal{T}$ in the space of $\mathcal{D}_2$. Let $o_{\mathbf{x}}$ be the data item that it encodes in $\mathcal{D}_1$. Then the *integration error* $\beta(\mathbf{x})$ of $\mathcal{T}$ is the distance between $\mathcal{T}(\mathbf{x})$ and $\texttt{emb}_2(o_{\mathbf{x}})$, *i.e.,* the imagined "ideal" vector that $\texttt{emb}_2$ would have generated in $\mathcal{D}_2$ if it were asked to embed $o_{\mathbf{x}}$. It should be noted that $\texttt{emb}_2(o_{\mathbf{x}})$ is a hypothetical vector that is not part of $\mathcal{D}_2$, but it resides in the same ambient space.

Note that the integration error β is defined for all vectors of $\mathcal{D}_1$, and reflects the divergence between the integrated vectors and their "ideal" position in the vector space of $\mathcal{D}_2$. As shown in Fig. 5, it is sensitive to how we group references $\mathcal{X}$ of $\mathcal{D}_1$ in step (1) of LA2M.

To reveal their connections, we further delve into the contributing factors of β. In particular, the integration error of $\mathbf{x}$, *i.e.,* $\|\mathcal{T}(\mathbf{x}) - \texttt{emb}_2(o_{\mathbf{x}})\|$, is the combined effect of two sources of errors:

- *Model disagreement*: the inherent difference between $\texttt{emb}_1$ and $\texttt{emb}_2$, *i.e.,* even if both models embed the same dataset, their produced vector databases would not align with each other.
- *Merging error*: the "generalization error" of isometry f_{iso}^{i} determined by references X_i for transforming and integrating vector $\mathbf{x}$ that are not in X_i.

Intuitively, model disagreement is inherent to embedding models and is hence constant *w.r.t.* any given $\mathcal{D}_1$ and $\mathcal{D}_2$. Computationally, this is reflected by that even the optimal isometry f_{iso}^{i} cannot exactly align all references in X_i of $\mathcal{D}_1$ to their images in Y_i of $\mathcal{D}_2$ by step (2) of LA2M. Hence, the optimal

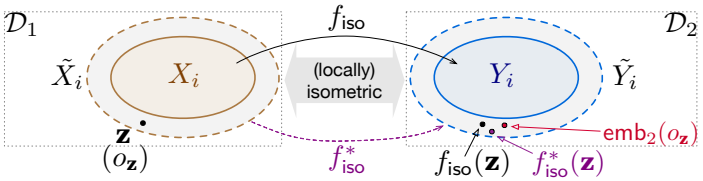


Fig. 6. Errors when using f_{iso} deduced from reference $X_i \to Y_i$ to map vector $\mathbf{z} \notin X_i$. (1) $f_{\text{iso}}(\mathbf{z})$: mapping of $\mathbf{z}$ by our isometry f_{iso}; (2) $f^*_{\text{iso}}(\mathbf{z})$: $\mathbf{z}$ mapped by the *optimal* isometry f^*_{iso} that agrees with our f_{iso} on X_i. (3) $\texttt{emb}_2(o_{\mathbf{z}})$: the envisaged embedding of $o_{\mathbf{z}}$ by $\texttt{emb}_2$, where $o_{\mathbf{z}}$ is the data item encoded by $\texttt{emb}_1$ in $\mathcal{D}_1$. The distance $\|(1)-(2)\| = \|f_{\text{iso}}(\mathbf{z}) - f^*_{\text{iso}}(\mathbf{z})\|$ corresponds to the merging error (model-independent), while $\|(1)-(3)\| = \|f_{\text{iso}}(\mathbf{z}) - \texttt{emb}_2(o_{\mathbf{z}})\|$ is the integration error (depends on $\texttt{emb}_1$ and $\texttt{emb}_2$).

alignment error of Equation 1 in step (2) of LA2M can be an indicator that helps decide if $\mathcal{D}_1$ and $\mathcal{D}_2$ can be integrated: if there are clusters X_i that cannot be aligned with Y_i with small alignment error, $\mathcal{D}_1$ and $\mathcal{D}_2$ may capture different similarity or irrelevant datasets and thus should not be integrated.

On the other hand, merging error originates from the way we integrate $\mathcal{D}_1$ and $\mathcal{D}_2$ and is what we aim to minimize in LA2M. We next define the notion and deduce an upper bound of the error.

Bounding merging error. As depicted in Fig. 6, the merging error of a non-reference $\mathbf{z}$, denoted by $\delta(\mathbf{z})$, is the distance between $f_{\text{iso}}(\mathbf{z})$ and $f^*_{\text{iso}}(\mathbf{z})$ in the vector space of $\mathcal{D}_2$, *i.e.,* $\|f_{\text{iso}}(\mathbf{z}) - f^*_{\text{iso}}(\mathbf{z})\|$, where

(a) f_{iso} is the isometry determined by LA2M over X_i and Y_i, *i.e.,* the optimal isometry that aligns local references X_i of $\mathcal{D}_1$ to Y_i of $\mathcal{D}_2$ with minimum sum of squared errors (*i.e.,* Equation 1);

(b) f^*_{iso} is the optimal isometry that aligns larger locally isometric regions $\tilde{X}_i$ and $\tilde{Y}_i$ of $\mathcal{D}_1$ and $\mathcal{D}_2$ vector spaces, where $\tilde{X}_i$ consists of reference vectors of X_i and non-reference vectors $\mathbf{z}$ mapped by f_{iso} (in step (3) of LA2M), and $\tilde{Y}_i = Y_i \cup \{\texttt{emb}_2(\mathbf{z}) \mid \mathbf{z} \in \tilde{X}_i \setminus X_i\}$.

Intuitively, f_{iso} is the optimal isometry that LA2M computes in step (2) via references X_i and Y_i, and f^*_{iso} is the optimal isometry that would be determined by mapping the larger reference sets $\tilde{X}_i$ and $\tilde{Y}_i$. Note that $\tilde{Y}_i$ contains hypothetical vectors that $\texttt{emb}_2$ would generate for data items encoded by non-reference vectors integrated by f_{iso} in step (3) of LA2M.

We next state our main result, an upper bound on $\delta(\mathbf{z})$ in terms of (a) the alignment errors of f_{iso} and f^*_{iso} on references X_i and Y_i, and (b) the distance between $\mathbf{z}$ and the centroid $\mathbf{c}_X$ of X_i.

More specifically, the *alignment error* of f_{iso} on X_i and Y_i is the total alignment errors of reference vectors in X_i *w.r.t.* f_{iso}, calculated by $\sum_{\mathbf{x}\in X_i} \|f_{\text{iso}}(\mathbf{x}) - \mathbf{y}\|^2$ (Equation 1), where $\mathbf{x} \in X_i$ corresponds to $\mathbf{y} \in Y_i$, *i.e.,* they encode the same data item; similarly for f^*_{iso}. Denote by α and α^* the alignment errors of f_{iso} and f^*_{iso} on (X_i, Y_i) respectively. For simplicity, in the sequel we write X_i (resp. Y_i) as X (resp. Y). We also treat X and Y as matrices with their vectors as columns.

Theorem 1: *Let $\sigma_{\min}(X)$ be the minimum singular value of X. Assume that X and $\tilde{X}$ (resp. Y and $\tilde{Y}$) have the same centroid. Then:*

$$\delta(\mathbf{z}) \le \frac{\alpha + \alpha^*}{\sigma_{\min}(X)} \|\mathbf{z} - \mathbf{c}_X + \mathbf{c}_Y\|$$

□

Proof: From Section 3.2 we know that $f_{\text{iso}}(\mathbf{x}) = s * R(\mathbf{x} - \mathbf{c}_X + \mathbf{c}_Y)$ for some rotation matrix R and scaling factor s; similarly $f^*_{\text{iso}}(\mathbf{x}) = s^* R^*(\mathbf{x} - \mathbf{c}_{\tilde{X}} + \mathbf{c}_{\tilde{Y}})$ for some R^* and s^*. Since X and $\tilde{X}$ have overlapping centriod, $\mathbf{c}_X = \mathbf{c}_{\tilde{X}}$; similarly, $\mathbf{c}_Y = \mathbf{c}_{\tilde{Y}}$. Denote by $\hat{X}$ the matrix translated from X (as a matrix) with translation vector $\mathbf{c}_Y - \mathbf{c}_X$.

Let $E = \sqrt{\sum_{\mathbf{x}\in X} \|f_{\text{iso}}(\mathbf{x}) - f^*_{\text{iso}}(\mathbf{x})\|^2}$. Written in matrix form,

$$E = \|(sR - s^*R^*)\hat{X}\|_F \quad \text{(Frobenius norm)}$$

Let singular value decomposition (SVD) of $\hat{X}$ be $\hat{X} = U\Sigma V^\top$ with

$$\Sigma = \text{diag}(\sigma_1, \ldots, \sigma_d), \quad \sigma_1 \geq \cdots \geq \sigma_d \geq 0,$$

and $V = [v_1, \ldots, v_d]$ containing the right singular vectors. Since U is orthogonal, $E = \|(s\,R - s^*R^*)\hat{X}\|_F = \|\Sigma\, V^\top (s\,R - s^*R^*)\|_F$. Taking the square and expanding the Frobenius norm, we have

$$\|\Sigma\, V^\top (s\,R - s^*R^*)\|_F^2 = \sum_{i=1}^{d} \sigma_i^2 \, \|v_i^\top (s\,R - s^*R^*)\|^2.$$

Hence, for each right singular direction v_i,

$$\|\, s\,Rv_i - s^*Rv_i\| \leq \frac{\|(s\,R - s^*R^*)X\|_F}{\sigma_i} \leq \frac{E}{\sigma_{\min}(\hat{X})} = \frac{E}{\sigma_{\min}(X)},$$

where $\sigma_{\min}(\hat{X}) = \min_i \sigma_i$ and $\sigma_{\min}(X)$ is the minimum non-zero singular of X (note that $\hat{X}$ and X have the same singular values).

Let $\mathbf{z} \in \mathbb{R}^d$ be a new point in $\tilde{X} \setminus X$. Express its translated version $\hat{\mathbf{z}} = \mathbf{z} + \mathbf{c}_Y - \mathbf{c}_X$ in the basis $\{v_1, \ldots, v_d\}$ as $\hat{\mathbf{z}} = \sum_{i=1}^{d} \alpha_i\, v_i$ with $\alpha_i = v_i^\top \hat{\mathbf{z}}$. Then, its images by f^*_{iso} and f_{iso} in $\mathcal{D}_2$ vector space are

$$f^*_{\text{iso}}(\mathbf{z}) = s^*R^*\hat{\mathbf{z}} = \sum_{i=1}^{d} \alpha_i \left[s^*Rv_i\right],\; f_{\text{iso}}(\mathbf{z}) = s\,R\hat{\mathbf{z}} = \sum_{i=1}^{d} \alpha_i \left[s\,Rv_i\right].$$

Thus, their difference is given by

$$s^*R^*\hat{\mathbf{z}} - s\,R\hat{\mathbf{z}} = \sum_{i=1}^{d} \alpha_i \left[s^*Rv_i - s\,Rv_i\right].$$

Taking norms and applying the triangle inequality,

$$\|s^*R^*\hat{\mathbf{z}} - s\,R\hat{\mathbf{z}}\| \leq \sum_{i=1}^{d} |\alpha_i|\, \|s^*Rv_i - s\,Rv_i\| \leq \frac{E}{\sigma_{\min}(X)} \sum_{i=1}^{d} |\alpha_i|.$$

By an application of the Cauchy–Schwarz inequality, we know $\sum_{i=1}^{d} |\alpha_i| \leq \sqrt{d}\, \|\hat{\mathbf{z}}\|$. Thus, we may write a succinct bound:

$$\|f^*_{\text{iso}}(\mathbf{z}) - f_{\text{iso}}(\mathbf{z})\| = \|s^*R^*\hat{\mathbf{z}} - s\,R\hat{\mathbf{z}}\| \leq \frac{E}{\sigma_{\min}(X)}\, \|\mathbf{z} - \mathbf{c}_X + \mathbf{c}_Y\|.$$

By the triangle inequality of Frobenius norm, we further have

$$E \leq \|sR\hat{X} - Y\|_F + \|s^*R^*\hat{X} - Y\|_F = \alpha + \alpha^*.$$

Putting together, we have $\delta(\mathbf{z}) \leq \dfrac{\alpha + \alpha^*}{\sigma_{\min}(X)} \|\mathbf{z} - \mathbf{c}_X + \mathbf{c}_Y\|$. □

Remark. To simplify the discussion, the upper bound assumes that X is full rank, *i.e.,* its singular values are all non-zero. This suffices to offer insights to inform the design specification of `LA2M`. As will be shown in Section 6, our design of `LA2M` works with both full rank and rank-deficient X.

Drawing insight from bound. Theorem 1 tells us the following.

(**I1**) Minimizing the alignment error α of f_{iso} over references X and Y helps reduce the merging error of non-reference vectors. This also justifies the objective function of Equation 1 and its application in step (2) of `LA2M`. As shown in Fig. 5 (blue line), the smaller X is, the lower α could be.

(**I2**) However, simply reducing X does not work. When X is too small, the minimum singular value $\sigma_{\min}(X)$ of its matrix form would be extremely small or even 0, which means $\delta(\mathbf{z})$ can be unbounded or infinitely large, despite that α is small. This is consistent with Fig. 5 where the overall integration error β is the worst when n approaches its maximum 1000, despite that α reaches its minimum.

(**I3**) This further tells us that poor conditioning of X (*i.e.,* small $\sigma_{\min}(X)$) leads to significantly amplified merging error. Moreover, directions of X with low variance (*i.e.,* small σ_i) are particularly susceptible to rotation ambiguity that elevates $\delta(z)$ hyperbolically.

(**I4**) The merging error $\delta(\mathbf{z})$ is also related to the position of $\mathbf{z}$ relative to the centroids of X and Y. The translation of X into $\hat{X}$ in the proof of Theorem 1 also implies that, if we center X and Y to their centriods first, then we would not need translation in f_{iso} and f^*_{iso}. Moreover, the upper bound of $\delta(\mathbf{z})$ would be then simplified to $\delta(\mathbf{z}) \leq \frac{\alpha+\alpha^*}{\sigma_{\min}(X)}\|\mathbf{z}\|$. As such, the closer $\mathbf{z}$ is to the translated centroid of X (*i.e.,* the origin) the smaller the merging error. In fact, this is exactly how we implemented step (2) of `LA2M`. This also justifies why we pick $f^{j^*}_{\text{iso}}$ that has the closest centroid when we merge a new non-reference vector $\mathbf{z}$ in step (3) of `LA2M`.

6 Informing the Framework Design

We next present the design choices of `LA2M` based on the insights offered by the bound of Section 5.

6.1 Breaking down the Global Isometry

We first consider step (1) of `LA2M`, by studying how we break down $\mathcal{X}$ into local groups to determine the local isometries of step (2). We aim to minimize the merging error in step (3) of `LA2M`.

Locality Vs. **dimensionality**. Consider a non-reference vector $\mathbf{z}$ of $\mathcal{D}_1$. According to (**I4**) of Section 5, to merge $\mathbf{z}$ into $\mathcal{D}_2$ with a minimum merging error, we shall use an isometry based on references X with a centroid as close to $\mathbf{z}$ as possible. By (**I1**), X should be as small as possible to minimize alignment error α. Further according to **I2** and **I3**, this should not make X rank-deficient, *i.e.,* the size $|X|$ should be at least d, where d is the dimensionality of the embeddings. These suggest that the ideal X for merging $\mathbf{z}$ would consists of at least d reference vectors of $\mathcal{D}_1$ that are closest to $\mathbf{z}$.

However, real-life embedding vectors often have high dimensionality. For instance, `OpenAI-Ada` embeddings are of 1536 dimension and that of `Mistral` is 1024. Consequently, this could necessitate large X and may inadvertently result in significant alignment errors. These, in turn, could lead to suboptimal merging errors. In addition, even if we have a large full rank X, its size might lead to nonzero but rather small singular values, exacerbating merging errors further.

Indeed, (**I3**) naturally motivates us to consider reducing the dimensionality of vectors via PCA (Principal Component Analysis [2]) to avoid poor conditioning of X, in particular eliminating directions with low variance (small singular values σ_i). This would give us an isometry f_{iso} based on X that is insusceptible to embedding noise.

Thus, determining the size of X reduces to finding the right dimensionality for the PCA-reduced reference vectors. Recall that to allow step (2) of `LA2M` to determine a local isometry from X to Y that can accurately bind vectors of $\mathcal{D}_1$ that are close to X, one has to assure X contains vectors that are able to determine inherent (*i.e.,* minimum) dimensionality of dissimilarity among data items encoded by X as well as those would be transformed by the local isometry determined by X. While we cannot exactly know the inherent dimensions, we however could approximate it with *intrinsic dimension* estimation over the reference vectors using Maximum Likelihood Estimation (MLE) [51].

The intrinsic dimension characterizes the minimal number of dimensions necessary to effectively represent manifold structures of data without significant information loss. The MLE-based intrinsic

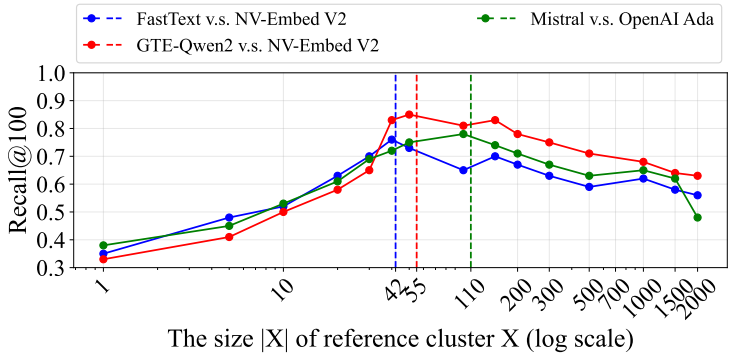


Fig. 7. Integration accuracy (recall@100) Vs **size of local reference clusters** X **over** SciFact**:** the vertical dashed lines mark the intrinsic dimensions. Benchmark results verify that the accuracy (recall@100) peaks when $|X|$ is close to the intrinsic dimension in all three integration cases.

dimension estimator determines this optimal intrinsic dimensionality of a given data point (vector) by analyzing its local neighborhoods. For each data point, the local intrinsic dimension is computed by examining the distances to its k-nearest neighbors. The intrinsic dimensionality $\hat{d}$ can be obtained via the MLE estimator[51]: $\hat{d} = \left[\frac{1}{N(K-1)} \sum_{i=1}^{N} \sum_{j=1}^{K-1} \log \frac{T_K(x_i)}{T_j(x_i)}\right]^{-1}$, where $T_j(x_i)$ denotes the distance from data point x_i to its j-th nearest neighbor, and N is the total number of data points of concern.

Mapping to our case, we set N to 10 and K to 20 for each dataset, and use $\hat{d}$ of the reference vector $\mathbf{x}$ closest to $\mathbf{z}$ to determine the size of X. In cases where we only know reference vectors, we use the average intrinsic dimensionality of randomly sampled reference vectors as a database-wide configuration of the size of X. Using benchmarks, results in Fig. 7 confirm that intrinsic dimension does serve as a quite accurate estimation of the best size of X for determining local isometries.

Computing with rank-deficient references. While we can use intrinsic dimension of reference vectors to decide the size of reference cluster X, it however requires us to reduce the dimensions of embedding vectors. To avoid directly altering vector databases for merging, we improve upon A2M with PCA in step (2) of LA2M, such that it works with small X containing high-dimensional embedding vectors, *i.e.,* rank-deficient reference clusters X.

Consider a pair of corresponding reference clusters $(X, Y) \in \mathbb{R}^{m \times d}$. We first perform SVD on X and Y, yielding $\text{SVD}(X) = U_X \Sigma_X V_X^\top$ and $\text{SVD}(Y) = U_Y \Sigma_Y V_Y^\top$. We then project X and Y into their respective $\hat{d}$-dimensional subspaces: $\hat{X} = XV_X^{\hat{d}}$, $\hat{Y} = YV_Y^{\hat{d}}$, where $\hat{d}$ is the size of X (*i.e.,* intrinsic dimensionality determined above), $V_X^{\hat{d}}, V_Y^{\hat{d}} \in \mathbb{R}^{d \times \hat{d}}$ are the top $\hat{d}$ right singular vectors of X and Y, respectively. We rewrite the objective function in Equation 1 as:

$$\arg\min_{s,R,t} \left\| sR(\hat{X} + t) - \hat{Y} \right\|_F^2. \tag{2}$$

Let $\hat{f}_{\text{iso}}$ be the optimal isometry determined by $\hat{X}$ and $\hat{Y}$. For a new vector $\mathbf{z} \in \mathbb{R}^{1 \times d}$, we first reduce its dimensionality using $V_X^{d'}$, yielding $\hat{\mathbf{z}} = \mathbf{z}V_X^{d'}$. We then apply $\hat{f}_{\text{iso}}$ to $\hat{\mathbf{z}}$ and finally reconstruct the transformed vector by multiplying with $V_Y^{d'\top}$: $\hat{f}_{\text{iso}}(\hat{\mathbf{z}})V_Y^{d'\top}$.

Online integration. Recall from Section 4.2 that steps (1) and (2) of LA2M work on reference vectors, independent of non-reference vectors of $\mathcal{D}_1$ that need to be merged into $\mathcal{D}_2$. In cases where there are constantly new vectors added to $\mathcal{D}_1$, LA2M separates step (3) from the first two steps, to reduce the time of merging these new vectors.

Table 2. Statistics of benchmarks

Dataset	#Train Pairs	#Test Queries	#Test Corpus Size
SciFact [80]	920	300	5,183
NFCorpus [9]	110,575	323	3,633
ArguAna [79]	n/a	1,406	8,674
SciDocs [18]	n/a	1,000	25,657
FiQA-2018 [59]	14,166	648	57,638

Specifically, as offline preprocessing, LA2M carries out steps (1) and (2) and materializes local isometries $f_{\text{iso}}^1, \ldots, f_{\text{iso}}^n$ as a vector database itself, where each vector is the centroid of a reference cluster X_i in $\mathcal{D}_1$, of which the encoded datum is the isometry f_{iso}^i.

LA2M merges non-reference vectors of $\mathcal{D}_1$ as an online process. Once a non-reference vector $\mathbf{x}$ is available or newly added to $\mathcal{D}_1$, LA2M finds the closest centroid $\mathbf{c}_j$ via a top-1 vector search over the centroids, and retrieves the corresponding isometry f_{iso}^i to merge $\mathbf{x}$.

To minimize merging error of new vectors of $\mathcal{D}_1$, we compute and materialize an isometry for each reference vector $\mathbf{x}$ in $\mathcal{X}$ of $\mathcal{D}_1$: we first find $\hat{d}$ nearest neighboring references of $\mathbf{x}$ in $\mathcal{X}$, forming a cluster X with $\mathbf{x}$ approximately as its centriod. We then compute the isometry f_{iso}^X determined by X. Hence, we have $n = |\mathcal{X}|$ number of isometries, each indexed by a reference vector $\mathbf{x} \in \mathcal{X}$ as its approximate centroid. In case we want to limit the number of materialized isometries, we iteratively merge pair of closest isometries:we take the union of their reference clusters and replace the pair of isometries with the one determined by their combined reference vectors.

6.2 Making Embeddings Mergeable

We next discuss the availability of reference vectors, *i.e.,* pairs of vectors $(\mathbf{x}, \mathbf{y})$ such that $\mathbf{x} \in \mathcal{D}_1$ and $\mathbf{y} \in \mathcal{D}_2$ encode the same data item. Naturally, known correspondence in embedding backfilling, entity linkage and domain knowledge are viable sources of reference vectors across datasets. However, in case they are not given as a priori, one may still benefit from LA2M by actively generating reference vectors with limited access to the embedding models.

More specifically, let $\mathcal{D}$ be a vector database that encode data items of $\mathcal{O}$. Assume that we need to integrate $\mathcal{D}$ into another (possibly unknown) vector database that offers access to their embedding model `emb`. We consider the task of selecting up to $\gamma * |\mathcal{D}|$ data items of $\mathcal{O}$ (vectors of $\mathcal{D}$) as reference items with some $\gamma \in (0, 1]$, to form reference vector pairs by requesting their encoding of `emb`.

By (**I4**) of Section 5, we know that ideally these $\gamma|\mathcal{D}|$ reference items shall cover all items of $\mathcal{O}$ with close proximity in $\mathcal{D}$. In light of this, we propose a reference sampling method that operates on $\mathcal{D}$. It is parameterized by L, the number of local isometries we want to generate, and works as follows.

(1) It first forms L clusters of $\mathcal{D}$ via k-means clustering.

(2) For each cluster C_i, it uniformly samples $\gamma|C_i|$ vectors as references. In total this samples a set $\mathcal{X}$ of $\gamma|\mathcal{D}|$ vectors from $\mathcal{D}$.

(3) It queries `emb` with the data items of $\mathcal{O}$ encoded by $\mathcal{X}$, yielding a set $\mathcal{Y}$ of corresponding vectors in the target vector database. The pair $(\mathcal{X}, \mathcal{Y})$ forms references that enable us to integrate $\mathcal{D}$ into the targeted vector database with LA2M.

Remark. For cases where one wants to avoid revealing any item in $\mathcal{O}$, one may use synthesized objects as the references. It however works best if $\mathcal{D}$ is generated by , *e.g.,* a diffusion embedding model which allows us to synthesize reference items that can cover $\mathcal{O}$ well.

Table 3. Embedding models

Methods	Scale	Require Training	Dimension
`GloVe` [68]	Small	Required	300
`FastText` [8]	Small	Required	300
`NV-Embed-V2` [50]	Medium	Optional	1024
`GTE-Qwen2` [55]	Medium	Optional	1024
`OpenAI-Ada` [66]	Large	Not Required	1536
`Mistral` [43]	Large	Not Required	1024

7 Experimental Study

Using benchmarks, we evaluated the quality of integrated vector databases across embedding models. We first specify the setup (Section 7.1). We then report integration results across 6 pairs of major embedding models over 5 text retrieval benchmarks (Section 7.2). We also present more evidence for the local isometry hypothesis (Section 7.3) and a case study on image benchmark (Section 7.4).

7.1 Experimental Setup

Benchmarks. We used 5 text publicly accessible retrieval benchmarks ranging from biomedical, open-domain, citation prediction and fact checking, as summarized in Table 2. Each benchmark consists of (a) a collection of documents and (b) a set of built-in query-answer pairs for evaluating similarity search. Each query is a text string and its answer is one of multiple documents in the dataset that are manually validated relevant (*i.e.,* similar) by the benchmark creators.

Embedding models. We used 6 popular embedding models as summarized in Table 3, ranging from small to large in scale. We used these embedding models to encode the text benchmarks as vectors so as to implement similarity search via vector search over the embedded vector databases.

Methods. Given two vector databases $\mathcal{D}_1$ and $\mathcal{D}_2$ embedded by different models, we implemented 5 methods to integrate $\mathcal{D}_1$ into $\mathcal{D}_2$:

- `Union`: directly taking the union $\mathcal{D}_1 \cup \mathcal{D}_2$.
- `A2M`: baseline introduced in Section 3.2, based on Hypothesis A.
- $\texttt{A2M}_{\texttt{MLP}}$: variant of `A2M` that learns a 2-layer multilayer perceptron (MLP) neural network [36] in place of the SVD-based isometry; it is optimized using Adam optimizer with a learning rate of 0.001 for 300 epochs and with the alignment error as training objective.
- `CCA`: a method that uses two global linear projections to jointly maximize correlation between the source and target embedding spaces, by following [56].
- `GW`: an approach that uses Gromov–Wasserstein distance [4] to directly learn an alignment of references without their explicit correspondence.
- `LA2M`: the `LA2M` framework of Sections 4.2, based on Hypothesis B.
- $\texttt{LA2M}_{\texttt{MLP}}$: a variant of `LA2M` that uses $\texttt{A2M}_{\texttt{MLP}}$ in step (2).

Note that, all these methods are data-driven in the sense that they integrate $\mathcal{D}_1$ into $\mathcal{D}_2$ without accessing to encoded data items or the embedding models. This assures that their comparison is fair.

Evaluation plan. For each benchmark of Table 2, by default we randomly partitioned its dataset into three subsets O_1, $O_\cap$ and O_2; we also randomly distributed the built-in answers evenly in O_1 and O_2. We then used a pair of embedding models $\texttt{emb}_1$ and $\texttt{emb}_2$ from Table 3 to embed $O_1 \cup O_\cap$ and $O_2 \cup O_\cap$ respectively into vector databases $\mathcal{D}_1$ and $\mathcal{D}_2$. For each integration method $\mathcal{A}$ above, we evaluated

Table 4. Integrating `Mistral` into `OpenAI-Ada` vectors

Notations: **R**: recall@100 (%); **R1**: recall@100[$\mathcal{D}_1$] (%); **RN:** Relative NDCG (%); ↑: higher the better; ↓: lower the better

method	SciFact				NFCorpus				ArguAna				SciDocs				FiQA-2018			
	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)
Union	32.00	0.00	8.93	1.42	15.17	0.00	16.28	1.43	28.19	0.00	8.74	1.42	17.60	0.00	11.13	1.42	27.78	0.00	6.96	1.42
A2M	27.00	6.86	10.04	0.96	10.84	7.26	17.80	0.93	29.91	3.09	9.71	1.02	12.60	4.02	18.63	0.99	25.31	0.00	7.02	1.05
GW	44.67	2.43	4.67	1.00	13.93	6.40	4.54	1.00	53.25	0.24	4.64	1.00	21.80	5.53	4.97	1.00	36.42	5.45	5.56	1.00
CCA	50.00	1.22	5.27	0.94	18.58	8.05	5.34	0.81	49.61	0.00	4.51	2.96	28.40	7.84	5.83	2.30	40.59	6.06	5.75	2.33
$A2M_{MLP}$	28.00	5.88	9.62	1.00	10.84	7.26	11.12	1.00	32.62	6.49	10.79	1.00	34.70	44.29	46.33	0.88	24.38	0.23	7.25	1.00
$LA2M_{MLP}$	35.00	24.02	13.85	0.83	13.62	14.10	11.72	0.81	39.61	23.95	15.03	0.87	14.20	12.90	27.55	0.86	16.20	6.31	12.12	0.88
LA2M	**76.67**	**86.76**	**36.53**	**0.75**	**34.06**	**28.21**	**30.82**	**0.36**	**79.23**	**82.14**	**32.46**	**0.81**	**42.00**	**54.23**	**52.05**	**0.68**	**58.02**	**75.23**	**41.37**	**0.28**

Table 5. Integrating `NV-Embed-V2` database into `OpenAI-Ada` database

method	SciFact				NFCorpus				ArguAna				SciDocs				FiQA-2018			
	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)
Union	32.00	0.00	8.93	1.41	14.24	0.00	16.41	1.41	28.19	0.00	8.74	1.41	16.70	0.00	10.98	1.41	27.78	0.00	6.95	1.42
A2M	32.00	0.00	8.94	1.26	14.24	0.00	17.20	1.24	28.19	0.00	8.76	1.29	16.70	0.00	9.45	1.28	27.78	0.00	6.96	1.28
GW	43.00	2.57	4.73	1.00	13.00	6.68	4.46	1.00	51.75	0.30	4.64	1.00	20.60	5.19	4.91	1.00	36.27	5.45	5.56	1.00
CCA	51.33	1.55	5.36	0.90	21.05	8.66	7.37	0.98	49.25	0.31	4.50	0.85	28.40	7.84	5.83	1.62	13.28	8.78	6.52	4.78
$A2M_{MLP}$	28.33	6.37	11.19	0.99	11.46	7.59	34.16	1.00	29.26	1.80	9.41	1.00	33.20	42.86	45.93	**0.29**	23.77	0.23	7.81	1.00
$LA2M_{MLP}$	39.33	30.88	16.28	0.83	15.48	16.03	26.64	0.81	39.19	23.45	15.40	0.87	13.20	12.76	26.33	0.86	15.43	4.21	12.03	0.88
LA2M	**79.33**	**90.69**	**49.81**	**0.72**	**35.29**	**40.93**	**37.22**	**0.39**	**85.22**	**90.22**	**41.96**	**0.47**	**41.10**	**54.23**	**53.68**	0.31	**58.49**	**75.93**	**42.27**	**0.28**

Table 6. Integrating `FastText` database into `OpenAI-Ada` database

method	SciFact				NFCorpus				ArguAna				SciDocs				FiQA-2018			
	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)
Union	25.33	0.45	8.84	1.21	13.93	2.97	16.06	1.22	29.55	0.00	8.66	1.23	12.20	2.40	11.22	1.13	26.23	0.23	6.96	1.17
A2M	16.67	4.91	10.01	0.88	10.53	8.90	22.50	0.85	25.34	1.42	9.26	0.90	6.70	0.14	14.78	0.93	15.43	0.23	8.00	0.93
GW	42.00	2.03	4.66	1.00	10.53	6.54	4.56	1.00	51.46	8.27	4.64	1.00	19.80	5.36	4.94	1.00	37.04	5.48	5.56	1.00
CCA	47.67	1.89	4.75	0.86	14.24	6.27	4.18	0.73	50.61	10.42	4.62	0.91	28.40	7.84	5.83	1.16	40.59	6.06	5.75	1.42
$A2M_{MLP}$	21.00	5.36	9.62	1.00	8.67	3.39	23.36	1.00	33.76	6.70	10.79	1.00	23.40	28.65	18.75	**0.29**	23.92	0.23	7.24	1.00
$LA2M_{MLP}$	35.33	31.70	14.62	0.83	12.07	11.44	27.46	0.81	33.26	14.42	12.16	0.87	11.20	8.74	24.70	0.86	14.20	2.29	10.69	0.89
LA2M	**67.00**	**74.55**	**26.33**	**0.77**	**26.63**	**25.85**	**29.04**	**0.76**	**68.24**	**55.43**	**25.89**	**0.82**	**39.30**	**32.13**	**28.23**	0.39	**50.93**	**43.71**	**23.64**	**0.29**

the quality of its integrated database $\mathcal{A}(\mathcal{D}_1, \mathcal{D}_2)$ by testing the benchmark queries. Specifically, for each benchmark query-answer pair (q, ans_q), we used top-k vector search with query vector $\texttt{emb}_2(q)$ over $\mathcal{A}(\mathcal{D}_1, \mathcal{D}_2)$ to retrieval answers. With the built-in ground-truth ans_q, we can quantify the quality of $\mathcal{A}(\mathcal{D}_1, \mathcal{D}_2)$ via search accuracy of $\texttt{emb}_2(q)$ with measures defined as follows.

Measures. We used 4 measures to quantify the quality of search results over the integrated database. Let Q denote the set of benchmark queries; the measures are defined as follows.

(a) *Recall@k*. For each benchmark query-answer pair $(q, \text{ans}_q) \in Q$, let $\text{rank}(\texttt{emb}_2(q), \mathcal{A}(\mathcal{D}_1, \mathcal{D}_2))$ be the rank of the top retrieved ans_q document by top-k vector search with $\texttt{emb}_2(q)$ over $\mathcal{A}(\mathcal{D}_1, \mathcal{D}_2)$. Then recall@$k$ is defined as $\frac{1}{|Q|}\sum_{q\in Q}\mathbb{I}\Big(\text{rank}(\texttt{emb}_2(q), \mathcal{A}(\mathcal{D}_1, \mathcal{D}_2)) \le k\Big)$, where $\mathbb{I}(p)$ is the indicator function, *i.e.,* 1 if p evaluates *True* and 0 otherwise.

(b) *Recall@k*[$\mathcal{D}_1$]. Since half of the benchmark answers are distributed in $\mathcal{D}_2$, when computing recall@k, $\texttt{emb}_2(q)$ can score 1 for half of the benchmark queries without the need of integrating $\mathcal{D}_1$. To eliminate such noise in recall@k, we also used recall@k[$\mathcal{D}_1$], which is the same as recall@k but considers Q that consists of (q, ans) where the top ans is encoded in $\mathcal{D}_1$ only.

(c) *Relative NDCG*. We also adopted NDCG [85], a popular accuracy measure that quantifies the relative rank changes of retrieved answers *w.r.t.* ground truth. Let $\mathcal{D}_* = \texttt{emb}_2(O_1 \cup O_\cap \cup O_2)$,

Table 7. Integrating `FastText` database into `NV-Embed-V2`

Notations: **R**: recall@100 (%); **R1**: recall@100[$\mathcal{D}_1$] (%); **RN:** Relative NDCG (%); ↑: higher the better; ↓: lower the better

method	SciFact				NFCorpus				ArguAna				SciDocs				FiQA-2018			
	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)
Union	26.33	2.23	8.83	1.21	13.62	5.46	14.08	1.23	30.34	0.91	8.65	1.24	10.80	2.36	10.46	1.15	27.32	0.69	6.95	1.17
A2M	18.33	4.91	9.95	0.95	8.05	5.46	15.57	0.95	25.91	2.23	9.27	0.96	5.70	1.53	16.44	0.96	16.51	1.37	8.03	0.93
GW	41.00	10.83	14.66	1.00	6.50	5.00	3.15	1.00	50.46	0.50	4.73	1.00	19.00	5.39	5.00	1.00	37.65	5.59	6.09	1.00
CCA	36.67	11.62	14.26	0.82	12.38	5.75	3.38	0.84	42.54	0.41	4.35	0.89	30.60	8.04	6.19	1.38	45.99	16.57	6.69	1.46
$A2M_{MLP}$	21.00	5.80	9.59	1.00	6.19	3.36	13.29	1.00	32.55	7.11	10.59	1.00	22.90	29.02	18.39	**0.31**	23.30	0.23	7.10	1.00
$LA2M_{MLP}$	34.67	29.02	13.61	0.91	9.29	7.56	16.57	0.91	34.33	15.63	12.14	0.93	11.30	10.71	24.39	0.90	53.12	66.23	28.18	0.35
LA2M	**76.67**	**84.82**	**28.78**	**0.85**	**34.06**	**24.37**	**20.83**	**0.43**	**64.60**	**60.00**	**21.92**	**0.50**	**39.30**	**32.13**	**31.29**	0.39	**58.02**	**75.23**	**41.37**	**0.28**

Table 8. Integrating `GTE-Qwen2` database into `NV-Embed-V2` database

method	SciFact				NFCorpus				ArguAna				SciDocs				FiQA-2018			
	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)
Union	31.67	0.00	8.95	1.41	18.27	0.00	14.09	1.41	28.48	0.00	8.65	1.41	19.60	0.00	9.91	1.41	28.86	0.00	6.94	1.41
A2M	32.33	1.46	8.92	1.21	15.48	5.08	13.34	1.20	28.48	0.00	8.65	1.31	18.40	1.27	10.78	1.22	28.86	0.00	6.93	1.30
GW	47.67	0.99	4.38	1.00	9.29	1.57	1.53	1.00	50.25	0.51	4.73	1.00	18.10	0.18	2.14	1.00	37.50	0.03	3.82	1.00
CCA	52.33	0.35	5.04	0.94	26.93	0.06	2.41	0.97	50.25	0.36	4.80	0.98	30.60	0.00	2.63	1.75	45.80	0.00	4.19	1.99
$A2M_{MLP}$	28.33	6.34	9.88	1.00	5.57	2.54	20.77	1.00	28.84	3.29	9.71	1.00	37.50	48.97	45.51	0.89	22.38	0.45	7.13	1.00
$LA2M_{MLP}$	43.33	34.63	15.46	0.90	14.55	14.41	18.06	0.90	39.83	24.45	13.81	0.93	15.30	17.09	27.49	0.71	13.73	5.00	9.80	0.89
LA2M	**85.33**	**95.12**	**50.53**	**0.45**	**31.27**	**36.86**	**23.54**	**0.44**	**86.65**	**92.02**	**40.42**	**0.40**	**44.60**	**60.17**	**51.89**	**0.30**	**57.87**	**70.45**	**31.35**	**0.29**

Table 9. Integrating `GloVe` into `FastText` vectors

method	SciFact				NFCorpus				ArguAna				SciDocs				FiQA-2018			
	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)	R(↑)	R1(↑)	RN(↑)	β(↓)
Union	19.67	0.00	11.94	2.76	8.36	0.00	26.79	2.90	19.34	0.00	14.42	3.45	9.30	0.00	17.18	2.93	12.19	4.23	10.59	0.62
A2M	19.67	0.00	16.29	0.30	8.36	0.00	49.71	0.27	19.34	0.00	21.24	0.23	11.30	0.00	42.27	0.22	12.19	3.29	18.95	0.24
GW	24.33	1.01	12.79	0.58	8.36	6.78	44.40	0.55	29.98	0.00	12.50	0.62	14.60	0.38	42.00	0.30	13.20	2.05	13.41	0.21
CCA	28.12	0.81	14.32	1.04	9.23	8.18	45.82	0.94	31.13	0.00	13.50	0.96	13.30	0.89	51.00	0.80	16.20	2.75	12.39	0.53
$A2M_{MLP}$	19.67	0.00	18.66	0.64	8.36	0.00	51.76	0.60	19.34	0.00	19.33	0.57	13.80	1.54	53.83	0.14	12.19	4.78	19.99	0.56
$LA2M_{MLP}$	19.33	0.47	25.44	0.15	8.36	0.00	51.97	0.14	18.70	2.61	27.96	0.12	13.40	0.56	45.61	0.14	17.13	4.19	33.28	0.28
LA2M	**39.00**	**31.16**	**41.57**	**0.10**	**11.15**	**2.99**	**63.91**	**0.10**	**40.40**	**38.19**	**41.28**	**0.09**	**14.90**	**10.39**	**58.12**	**0.11**	**18.20**	**8.53**	**36.11**	**0.04**

i.e., the complete ground-truth vector database that embeds the entire dataset with $\texttt{emb}_2$. For each benchmark (q, ans_q), denote by $r_{\text{merge}}(q)$ and $r_*(q)$ the rank of the first retrieved answer document of ans_q by $\texttt{emb}_2(q)$ over integrated database and over $\mathcal{D}_*$, respectively. The *relative NDCG score* $s(q)$ of q is $1/\log_2(|r_{\text{merge}}(q) - r_*| + 1)$. The relative NDCG of the integrated database is measured by the average score of all benchmark queries, *i.e.,* $\sum_{q\in Q} s(q)/|Q|$.

(d) *Integration error* β (Section 5). This measures, for each item $o \in O_1$, the distance $\|\mathcal{A}(\texttt{emb}_1(o)) - \texttt{emb}_2(o)\|$ in the $\mathcal{D}_2$ vector space, where $\mathcal{A}(\texttt{emb}_1(o))$ is the representation of o in the integrated database by $\mathcal{A}$. The average β over all data items in O_1 is reported.

Intuitively, higher score in measures (a)-(c) means better search quality over the integrated databases; for measure (d), however, the lower the better.

Configuration. All the experiments were run on a server that is equipped with an AMD EPYC 7713P@2GHz CPU with 120GB RAM and an NVIDIA A100 80GB GPU.

7.2 Quality of Integrated Vector Databases

Quality of integration. Using the evaluation plan of Section 7.1, we evaluated the search quality over the integrated vector database of all 5 benchmark datasets across 6 pairs of embedding models.

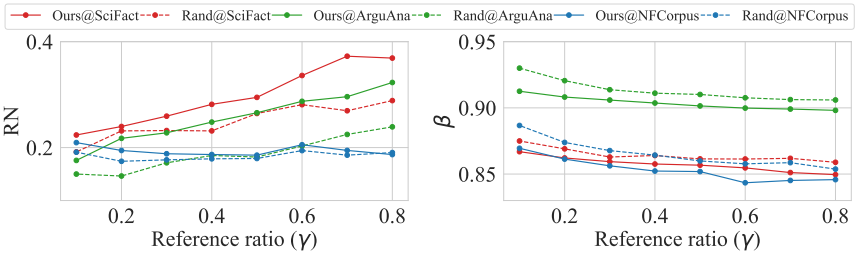


Fig. 8. Impact of reference vectors (`FastText`Vs `NV-Embed-V2`)

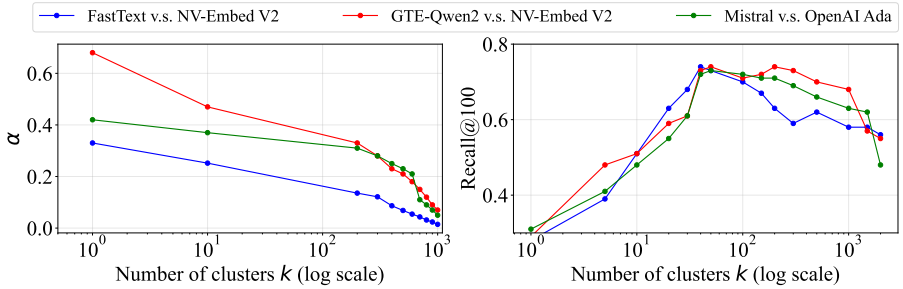


Fig. 9. Global Vs. **local isometries**

The results are reported in Tables 4-9, and tell us the following.

(1) `LA2M` consistently produces the best integrated vector databases over all benchmarks under all quality measures. For instance, its recall@100 is consistently more than twice of that of `Union`, despite that recall@100 is in strong favor of `Union` since it is guaranteed to retrieve all query answers that are distributed only in the target database by design and recall@100 gives them equal significance as those integrated from the source database. The results for recall@100$[\mathcal{D}_1]$ is even greater, with `Union` failed to retrieve any answers from $\mathcal{D}_1$ (*i.e.,* recall@100$[\mathcal{D}_1] = 0$) in most cases, while that of `LA2M` is *e.g.,* 90.69% when we integrate `NV-Embed-V2` into `OpenAI-Ada` databases. The results for relative NDCG and integration error β are similar.

(2) Specifically, `LA2M` consistently and significantly outperforms `A2M` in each and every case. This further demonstrates that the global isometry hypothesis does not hold for real-world datasets, whereas the local isometry hypothesis is however valid. It also confirms the effectiveness of local hypothesis and the `LA2M` framework for integrating vector databases based on this hypothesis.

(3) The use of isometry is instrumental, as shown by the gap between `LA2M` and $\texttt{LA2M}_{\texttt{MLP}}$ which replaces isometries with nonlinear neural network (MLP) in step (2) of `LA2M`. This is because isometries retain search results over $\mathcal{D}_1$ when integrating into $\mathcal{D}_2$, while nonlinear functions cannot despite that they may have smaller alignment error over references, *i.e.,* Theorem 1 does not hold for them.

(4) More advanced global alignment strategies like `CCA` and `GW` perform slightly better than vanilla `A2M` baselines. This is due to their structural modeling capabilities: `CCA` captures shared semantic dimensions via correlation maximization, while `GW` preserves global topology and relaxes the rigid point-to-point constraint. These lead to moderate gains in relative metrics such as relative NDCG.

However, both methods lack local consistency guarantees and are significantly outperformed by `LA2M` across all metrics. They do not preserve neighborhood structure, leading to poor recall@100$[\mathcal{D}_1]$ and increased sensitivity to geometric distortions. This highlights the importance of local isometries.

(5) Although `LA2M` achieves substantial relative improvements, its absolute recall remains bounded

by each embedding model's native performance. In addition to the errors caused by different integration methods, the integration error is also influenced by two external factors: (a) the intrinsic difficulty of specific benchmarks (e.g., `FastText` itself achieves only 17% Recall@100 on NFCorpus [77]), and (b) the inherent model disagreement between embedding models, as detailed in Section 5, which stems from differences in architecture, vocabulary coverage and training corpora between `GloVe` and `FastText`. As remarked in Section 2, these errors reflect the extent of inherent mergeability of these vector databases, regardless of the integration method.

Impact of references. We also evaluated the impact of reference vectors, by varying their ratio in the database, and by comparing our reference selection method (Section 6.2) with random uniform sampling. Figure 8 depicts the results of integrating `FastText` into `NV-Embed-V2` databases.

(1) In case no references are provided by the dataset, our reference selection method works effectively, consistently better than uniform sampling with the same number of references, *i.e.,* γ.

(2) Our approach is robust against varying number of reference vectors available, *e.g.,* its integration error increases just by 3.1% when the percentage γ of reference vectors drops from 80% to 20%. The results for other quality measures are similar.

7.3 **Global** Vs. **Local Isometry Hopythesis**

We consolidated the local isometry hypothesis (Section 4.1) with further evidence. Specifically, we evaluated the quality of integrated vector databases by a variant of `LA2M` that uses isometries produced by k-means clustering of the global isometry, with varying k. To do this, we created local isometries by first clustering references into k non-overlapping clusters via k-means clustering and then computing an isometry for each cluster by applying `A2M` (or step (2) of `LA2M`). Intuitively, a larger k means more clusters and thus local isometries of smaller scope; conversely, a smaller k indicates larger isometries. By varying k, we report in Fig. 9 the impact of the scope of local isometries on the quality of integrated database. The results tell us the following.

(1) The alignment error (α) over reference vectors consistently decreases with smaller clusters, *i.e.,* larger k. This does confirm that local structures in vector databases defined by shorter distances are more likely isometric across embedding models.

(2) However, isometries determined by vectors with too small inter-distances (extremely larger k) yield degraded integration quality, *e.g.,* recall@100. This is due to that the clusters would be too small to bind non-reference vectors for integration, confirming Theorem 1.

7.4 Case Study: Image Search

Privacy-preserving image search. Consider a hypothetical case study for image search across two parties, Alice and Bob, each holding part of the CIFAR-10 image dataset [47], partitioned as described in Section 7.1. Alice would like to search over the union of their image libraries collectively. However, Alice does not want to reveal her query image to Bob, nor is Bob willing to explicitly share his image files. Instead, Bob shares just the embedding vectors of his images produced by a local embedding model ResNet [37]. Alice then integrates her image embeddings (by DenseNet [39]) with those of Bob's. For each query image, Alice queries the integrated vector database. If the answer vector is from Bob, Alice then only needs to ask for this particular answer image and it is up to Bob to decide if he is willing to share the answer image with Alice. In the entire process, no image file is shared between Alice and Bob.

Results. The image search results for 10 query images (Fig. 10(a)) over the CIFAR-10 image dataset are given in Fig. 10. Compared to `Union`, with `LA2M` the image search results (via top-1 vector search)

(a) Alice's query image: 10 images from the CIFAR-10 dataset

(b) Top-1 image with `Union`: each corresponds to the query image directly above

(c) Top-1 image with `LA2M`: each corresponds to the query image above in (a)

Fig. 10. Privacy-preserving image search via vector integration

are more accurate. For instance, for all 10 query images, the search result with `LA2M` gave answer images consistently of the same category as the queries, while `Union` failed in 9 out of 10 occasions.

8 Additional Related Work

We have reviewed and compared with current research on vector databases in Section 1. Below, we extend our discussion to include related research in other areas. Although this does not directly address vector databases, it is technically relevant to our problem of integrating vector databases.

Embedding alignment. A related line of research is from the NLP community on aligning embeddings for (a) cross-lingual word translation and, more generally, (b) cross-domain adaptation.

Cross-lingual word embedding alignment. A major application of embedding alignment is cross-lingual dictionary induction, where words are represented as embeddings and cross-lingual word translation boils down to finding an alignment function that maps embeddings across languages [5, 44, 63, 74]. Mikolov et al. [63] first formulate this as a supervised regression problem, to learn a linear mapping by minimizing squared Euclidean distances between seed word pairs as supervised data. Building on this, subsequent studies add orthogonality constraints [74], optimize training loss objectives via Wasserstein or Gromov–Wasserstein distances [13], contrastive alignment [87], end-to-end learning [44] or Canonical Correlation Analysis (CCA) [5], and explore post-hoc adapters [48]. There have also been unsupervised variants that use adversarial training over frequent words [49] to generate initial seed word pairs, or directly use a permutation matrix in the training objective [30] to enumerate all correspondences between words across languages.

All these methods build on the key property of word embeddings discovered by Mikolov et al. [63]: word embeddings exhibit *global geometry similarity* across different languages, which allows for accurate alignment via a single transformation that preserves the global geometric structure. This foundation, however, does not hold in vector databases (Section 3), for two reasons: (a) Embeddings in vector databases, particularly those involving non-linguistic modalities, *e.g.,* images and sentence embeddings, lack the global geometric regularities of word embeddings in dictionaries. (b) Translation emphasizes more on aligning the distribution of word embeddings while vector databases target top-k search and our objective is to enable top-k retrieval across independently trained embeddings, which demands preservation of local structures rather than global correspondence. As a result, our work is based on the *local* isometry hypothesis (Section 4) as opposed to the global geometry similarity property of word embeddings. Our technical results center around the problem of deciding the optimal locality of the local isometries (Sections 5-6), to maximize the quality of integrated databases.

Cross-domain adaptation. Beyond cross-lingual word translation, similar idea, dubbed as cross-domain embedding alignment or adaptation [16, 54], has been exploited to augment training set and

improve model training performance. The key idea is to transform the embeddings in the training set to approximate the distributional structure of a target domain, via (a) metric learning to fit a domain-invariant Mahalanobis distance metric by minimizing information-theoretic divergences such as MMD [41, 82, 84], (b) manifold alignment to learn both domain-specific projections and sparse correspondences under the assumption of a shared low-dimensional manifold [12, 19, 70], and (c) adversarial transformation [46, 58, 62] to render transformed source embeddings indistinguishable from target embeddings in distributional terms via GAN-based training.

Nonetheless, these methods either cannot generalize beyond the known correspondence between source and target datasets *e.g.,* metric learning, or fail to bound per-point distortion, like manifold alignment and adversarial training. This is because they build on the same assumption of global source-target geometry consistency as seen in cross-lingual word translation. Consequently, they produce a single global alignment mapping that does not fit for vector databases that prioritize local geometry consistency for top-k search and generalized errors to data unknown to the alignment.

Domain adaptation goes beyond embedding adaptation by modifying models to address domain shifts without changing input data (see [53] for a survey). Key techniques include: (a) adversarial fine-tuning [57] to learn domain-invariant representations via adversarial objectives, (b) gradient reversal networks [27] that confuse the discriminator to create indistinguishable source and target features, and (c) asymmetric distribution alignment [88] to manage density imbalances in latent space. They all require access to the model architecture, labeled data, or intermediate feature representations.

However, these methods are unsuitable for vector integration, where neither model internals nor labeled examples are available. Moreover, they can only transform and adapt embedding vectors that are known before the construction (*e.g.,* training) of the adaption function, not any, potentially new, non-reference embedding vectors that our method (LA2M) merges into the target database. Even if we ignore such restrictions, domain adaptation still relies on global correlation between source and target domains, which makes it prone to the same limitations of embedding alignment seen above.

Data integration. Data integration offers a unified global schema for users to pose queries against multiple independent relational databases with varying schemas [21, 35]. It maintains global-to-local schema mappings, translating user queries into corresponding queries for local databases, and returning the aggregated query results. The term has also been used to describe the process of combining heterogeneous data sources, *e.g.,* omics data in life sciences [29]. In this work, we use the term to describe our initiative to integrate multiple vector databases from different embedding models.

9 Conclusion

In this work, we have presented our vision towards vector database integration across different embedding models. Realizing such vision is believed to benefit a few application scenarios, *e.g.,* embedding model iteration and vector back-filling, healthcare data federation, privacy-preserving similarity search, and federated graphRAG and clustering. We have developed a first attempt towards this vision, by both experimentally examining and theoretically characterizing the local isometry hypothesis for cross-model vector database integration. Benchmark results verify that our approach is able to integrate vector databases produced by major embedding models, including NV-embed-V2, OpenAI Ada, GloVe and Mistral, while offering high recall of top-k search over the integrated data.

We are currently incorporating unsupervised learning to automatically deduce reference vectors for our framework when they are not available. In addition, we also aim to study clustering over integrated vector databases without necessitating dataset integration.

Acknowledgments

This work is supported by RAEng RF\201920\19\319 and the Huawei-Edinburgh Joint Lab.

References

[1] Vector search features and limits. https://docs.aws.amazon.com/memorydb/latest/devguide/vector-search-limits.html, 2025.

[2] H. Abdi and L. J. Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[3] J. K. Aggarwal, Q. Cai, W. Liao, and B. Sabata. Articulated and elastic non-rigid motion: A review. In *Proceedings of 1994 IEEE Workshop on Motion of Non-rigid and Articulated Objects*, pages 2–14. IEEE, 1994.

[4] D. Alvarez-Melis and T. S. Jaakkola. Gromov-wasserstein alignment of word embedding spaces. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 1881–1890. Association for Computational Linguistics, 2018.

[5] M. Artetxe, G. Labaka, and E. Agirre. Learning principled bilingual mappings of word embeddings while preserving monolingual invariance. In J. Su, X. Carreras, and K. Duh, editors, *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, pages 2289–2294. The Association for Computational Linguistics, 2016.

[6] M. M. Astrahan, M. W. Blasgen, D. D. Chamberlin, K. P. Eswaran, J. N. Gray, P. P. Griffiths, W. F. King, R. A. Lorie, P. R. McJones, J. W. Mehl, G. R. Putzolu, I. L. Traiger, B. W. Wade, and V. Watson. System r: relational approach to database management. *ACM Trans. Database Syst.*, 1(2), June 1976.

[7] I. Azizi, K. Echihabi, and T. Palpanas. Graph-based vector search: An experimental evaluation of the state-of-the-art. *Proc. ACM Manag. Data*, 3(1), 2025.

[8] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*, 2016.

[9] V. Boteva, D. Gholipour, A. Sokolov, and S. Riezler. A full-text learning to rank dataset for medical information retrieval. In *Advances in Information Retrieval: 38th European Conference on IR Research, ECIR 2016, Padua, Italy, March 20–23, 2016. Proceedings 38*, pages 716–722. Springer, 2016.

[10] N. Bouziani, S. Tyagi, J. Fisher, J. Lehmann, and A. Pierleoni. REXEL: an end-to-end model for document-level relation extraction and entity linking. In *NAACL*, 2024.

[11] C. Chen, C. Jin, Y. Zhang, S. Podolsky, C. Wu, S. Wang, E. Hanson, Z. Sun, R. Walzer, and J. Wang. Singlestore-v: An integrated vector database system in singlestore. *Proc. VLDB Endow.*, 2024.

[12] D. Chen, B. Fan, C. G. Oliver, and K. M. Borgwardt. Unsupervised manifold alignment with joint multidimensional scaling. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

[13] L. Chen, Z. Gan, Y. Cheng, L. Li, L. Carin, and J. Liu. Graph optimal transport for cross-domain alignment. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 1542–1553. PMLR.

[14] L. J. Chen and Y. Papakonstantinou. Context-sensitive ranking for document retrieval. In *SIGMOD Conference*, pages 757–768. ACM, 2011.

[15] T. Chen, H. Yin, Y. Zheng, Z. Huang, Y. Wang, and M. Wang. Learning elastic embeddings for customizing on-device recommenders. In *KDD*, 2021.

[16] C. Chu and R. Wang. A survey of domain adaptation for neural machine translation. In E. M. Bender, L. Derczynski, and P. Isabelle, editors, *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1304–1319. Association for Computational Linguistics, 2018.

[17] N. C. Chung, B. Miasojedow, M. Startek, and A. Gambin. Jaccard/tanimoto similarity test and estimation methods for biological presence-absence data. *BMC Bioinform.*, 20-S(15):644, 2019.

[18] A. Cohan, S. Feldman, I. Beltagy, D. Downey, and D. S. Weld. Specter: Document-level representation learning using citation-informed transformers. *arXiv preprint arXiv:2004.07180*, 2020.

[19] Z. Cui, H. Chang, S. Shan, and X. Chen. Generalized unsupervised manifold alignment. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems 2014, December 8-13 2014, Montreal, Quebec, Canada*, pages 2429–2437.

[20] databricks. Improving retrieval and rag with embedding model finetuning. https://www.databricks.com/blog/improving-retrieval-and-rag-embedding-model-finetuning, 2025.

[21] A. Doan, A. Halevy, and Z. Ives. *Principles of data integration.* Elsevier, 2012.

[22] X. L. Dong. The journey to a knowledgeable assistant with retrieval-augmented generation (RAG). In *SIGMOD Conference Companion*, page 3. ACM, 2024.

[23] X. L. Dong and D. Srivastava. *Big data integration.* Morgan & Claypool Publishers, 2015.

[24] C. Eckart and G. Young. The approximation of one matrix by another of lower rank. *Psychometrika*, 1(3):211–218, 1936.

[25] W. Fan, Y. Ding, L. Ning, S. Wang, H. Li, D. Yin, T. Chua, and Q. Li. A survey on RAG meeting llms: Towards retrieval-augmented large language models. In *KDD*, pages 6491–6501. ACM, 2024.
[26] C. Foster, B. Sevilmis, and B. Kimia. Generalized relative neighborhood graph (grng) for similarity search. *Pattern Recognition Letters*, 188:103–110, 2025.
[27] Y. Ganin and V. S. Lempitsky. Unsupervised domain adaptation by backpropagation. In F. R. Bach and D. M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1180–1189. JMLR.org.
[28] B. Golshan, A. Halevy, G. Mihaila, and W.-C. Tan. Data integration: After the teenage years. In *PODS*, 2017.
[29] D. Gomez-Cabrero, I. Abugessaisa, D. Maier, A. E. Teschendorff, M. Merkenschlager, A. Gisel, E. Ballestar, E. Bongcam-Rudloff, A. Conesa, and J. Tegnér. Data integration in the era of omics: current and future challenges. *BMC Syst. Biol.*, 8(S-2):I1, 2014.
[30] E. Grave, A. Joulin, and Q. Berthet. Unsupervised alignment of embeddings with wasserstein procrustes. In *AISTATS*, pages 1880–1890, 2019.
[31] P. E. Green. *Mathematical tools for applied multivariate analysis*. Academic Press, 2014.
[32] R. Guo, X. Luan, L. Xiang, X. Yan, X. Yi, J. Luo, Q. Cheng, W. Xu, J. Luo, F. Liu, Z. Cao, Y. Qiao, T. Wang, B. Tang, and C. Xie. Manu: A cloud native vector database management system. *Proc. VLDB Endow.*, 15(12):3548–3561, 2022.
[33] S. Guo, Y. Ji, C. Zhang, C. Xu, and J. Xu. vcbir: A verifiable search engine for content-based image retrieval. In *ICDE*, pages 1730–1733. IEEE, 2020.
[34] S. Guo, J. Xu, C. Zhang, C. Xu, and T. Xiang. Imageproof: Enabling authentication for large-scale image retrieval. In *ICDE*, pages 1070–1081. IEEE, 2019.
[35] A. Y. Halevy, A. Rajaraman, and J. J. Ordille. Data integration: The teenage years. In *VLDB*, 2006.
[36] T. Hastie, R. Tibshirani, and J. H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009.
[37] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.
[38] W. Hu, R. Bansal, K. Cao, N. Rao, K. Subbian, and J. Leskovec. Learning backward compatible embeddings. In *KDD*, 2022.
[39] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017.
[40] G. Huang, Z. Liu, L. van der Maaten, and K. Q. Weinberger. Densely connected convolutional networks. In *CVPR*, pages 2261–2269, 2017.
[41] S. Ibrahimi, N. van Noord, Z. J. M. H. Geradts, and M. Worring. Deep metric learning for cross-domain fashion instance retrieval. In *2019 IEEE/CVF International Conference on Computer Vision Workshops, ICCV Workshops 2019, Seoul, Korea (South), October 27-28, 2019*, pages 3165–3168. IEEE, 2019.
[42] S. M. Islam, S. Joardar, and A. A. Sekh. A survey on fashion image retrieval. *ACM Comput. Surv.*, 56(6):140:1–140:25, 2024.
[43] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. de las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier, L. R. Lavaud, M.-A. Lachaux, P. Stock, T. Le Scao, T. Lavril, T. Wang, T. Lacroix, and W. El Sayed. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.
[44] A. Joulin, P. Bojanowski, T. Mikolov, H. Jégou, and E. Grave. Loss in translation: Learning bilingual word mapping with a retrieval criterion. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2979–2984. Association for Computational Linguistics, 2018.
[45] G. A. Kaissis, M. R. Makowski, D. Rückert, and R. F. Braren. Secure, privacy-preserving and federated machine learning in medical imaging. *Nature Machine Intelligence*, 2(6):305–311, 2020.
[46] N. Kondapaneni, M. Marks, M. Knott, R. Guimarães, and P. Perona. Text-image alignment for diffusion-based perception. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2024, Seattle, WA, USA, June 16-22, 2024*, pages 13883–13893. IEEE, 2024.
[47] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.
[48] S. Kulshreshtha, J. L. R. Garcia, and C.-Y. Chang. Cross-lingual alignment methods for multilingual BERT: A comparative study. In T. Cohn, Y. He, and Y. Liu, editors, *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 933–942. Association for Computational Linguistics, 2020.
[49] G. Lample, A. Conneau, M. Ranzato, L. Denoyer, and H. Jégou. Word translation without parallel data. In *ICLR*, 2018.
[50] C. Lee, R. Roy, M. Xu, J. Raiman, M. Shoeybi, B. Catanzaro, and W. Ping. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*, 2024.
[51] E. Levina and P. J. Bickel. Maximum likelihood estimation of intrinsic dimension. In *Advances in Neural Information Processing Systems 17 [Neural Information Processing Systems, NIPS 2004, December 13-18, 2004, Vancouver, British*

Columbia, Canada], pages 777–784, 2004.
[52] H. Li, T. N. Chan, M. L. Yiu, and N. Mamoulis. FEXIPRO: fast and exact inner product retrieval in recommender systems. In *SIGMOD Conference*, pages 835–850. ACM, 2017.
[53] J. Li, Z. Yu, Z. Du, L. Zhu, and H. T. Shen. A comprehensive survey on source-free domain adaptation. 46(8):5743–5762.
[54] J. Li, Z. Yu, Z. Du, L. Zhu, and H. T. Shen. A comprehensive survey on source-free domain adaptation. *IEEE Trans. Pattern Anal. Mach. Intell.*, 46(8):5743–5762, 2024.
[55] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang. Towards general text embeddings with multi-stage contrastive learning. *arXiv preprint arXiv:2308.03281*, 2023.
[56] A. Lu, W. Wang, M. Bansal, K. Gimpel, and K. Livescu. Deep multilingual correlation for improved word embeddings. In R. Mihalcea, J. Y. Chai, and A. Sarkar, editors, *NAACL HLT 2015, the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 250–256. The Association for Computational Linguistics, 2015.
[57] M. Lu, Z. Huang, Y. Zhao, Z. Tian, Y. Liu, and D. Li. DaMSTF: Domain adversarial learning enhanced meta self-training for domain adaptation. In A. Rogers, J. L. Boyd-Graber, and N. Okazaki, editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 1650–1668. Association for Computational Linguistics, 2023.
[58] J. Luo, Y. Zhao, X. Luo, Z. Xiao, W. Ju, L. Shen, D. Tao, and M. Zhang. Cross-domain diffusion with progressive alignment for efficient adaptive retrieval. *IEEE Transactions on Image Processing*, 34:1820–1834, 2025.
[59] M. Maia, S. Handschuh, A. Freitas, B. Davis, R. McDermott, M. Zarrouk, and A. Balahur. Www'18 open challenge: financial opinion mining and question answering. In *Companion proceedings of the the web conference 2018*, pages 1941–1942, 2018.
[60] Y. Malkov, A. Ponomarenko, A. Logvinov, and V. Krylov. Approximate nearest neighbor algorithm based on navigable small world graphs. *Inf. Syst.*, 45:61–68, 2014.
[61] Y. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Trans. Pattern Anal. Mach. Intell.*, 42(4):824–836, 2020.
[62] X. Mao, Q. Li, and H. Xie. AlignGAN: Learning to align cross-domain images with conditional generative adversarial networks. *Corr*, abs/1707.1400, 2017.
[63] T. Mikolov, Q. V. Le, and I. Sutskever. Exploiting similarities among languages for machine translation. *CoRR*, abs/1309.4168, 2013.
[64] R. J. Miller. Open data integration. *Proc. VLDB Endow.*, 2018.
[65] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers. MTEB: massive text embedding benchmark. In *EACL*, pages 2006–2029. Association for Computational Linguistics, 2023.
[66] OpenAI. New and improved embedding model: Text-embedding-ada-002. https://openai.com/blog/new-and-improved-embedding-model, 2022. Accessed: 2025-04-04.
[67] J. J. Pan, J. Wang, and G. Li. Survey of vector database management systems. *VLDB J.*, 33(5):1591–1615, 2024.
[68] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In A. Moschitti, B. Pang, and W. Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics.
[69] X. Perez-Sala, F. D. la Torre, L. Igual, S. Escalera, and C. Angulo. Subspace procrustes analysis. *International Journal of Computer Vision*, 121(3):327 – 343, February 2017.
[70] J. S. Rhodes and A. G. Rustad. Random forest-supervised manifold alignment. In W. Ding, C.-T. Lu, F. Wang, L. Di, K. Wu, J. Huan, R. Nambiar, J. Li, F. Ilievski, R. Baeza-Yates, and X. Hu, editors, *IEEE International Conference on Big Data, Bigdata 2024, Washington, DC, USA, December 15-18, 2024*, pages 3309–3312. IEEE, 2024.
[71] W. Shen, Y. Li, Y. Liu, J. Han, J. Wang, and X. Yuan. Entity linking meets deep learning: Techniques and solutions. *IEEE Trans. Knowl. Data Eng.*, 35(3), 2023.
[72] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In Y. Bengio and Y. LeCun, editors, *ICLR*, 2015.
[73] A. Singh, A. Ehtesham, S. Kumar, and T. T. Khoei. Agentic retrieval-augmented generation: A survey on agentic RAG. *CoRR*, abs/2501.09136, 2025.
[74] S. L. Smith, D. H. P. Turban, S. Hamblin, and N. Y. Hammerla. Offline bilingual word vectors, orthogonal transformations and the inverted softmax. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.
[75] C. Teflioudi, R. Gemulla, and O. Mykytiuk. LEMP: fast retrieval of large entries in a matrix product. In *SIGMOD Conference*, pages 107–122. ACM, 2015.
[76] H. Temiz, B. Gökberk, and L. Akarun. Multi-view reconstruction of 3d human pose with procrustes analysis. In *IPTA*, pages 1–5. IEEE, 2019.
[77] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych. Beir: A heterogenous benchmark for zero-shot

evaluation of information retrieval models. *arXiv preprint arXiv:2104.08663*, 2021.
[78] L. Van der Maaten and G. Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.
[79] H. Wachsmuth, S. Syed, and B. Stein. Retrieval of the best counterargument without prior topic knowledge. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 241–251, 2018.
[80] D. Wadden, S. Lin, K. Lo, L. L. Wang, M. van Zuylen, A. Cohan, and H. Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7534–7550, Online, Nov. 2020. Association for Computational Linguistics.
[81] Y. Wan, G. Zou, and B. Zhang. Composed image retrieval: a survey on recent research and development. *Appl. Intell.*, 55(6):482, 2025.
[82] H. Wang, W. Wang, C. Zhang, and F. Xu. Cross-domain metric learning based on information theory. In C. E. Brodley and P. Stone, editors, *Proceedings of the Twenty-eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, pages 2099–2105. AAAI Press.
[83] J. Wang, X. Yi, R. Guo, H. Jin, P. Xu, S. Li, X. Wang, X. Guo, C. Li, X. Xu, K. Yu, Y. Yuan, Y. Zou, J. Long, Y. Cai, Z. Li, Z. Zhang, Y. Mo, J. Gu, R. Jiang, Y. Wei, and C. Xie. Milvus: A purpose-built vector data management system. In *SIGMOD*, 2021.
[84] W. Wang, H. Wang, C. Zhang, and Y. Gao. Cross-domain metric and multiple kernel learning based on information theory. *Neural Computation*, 30(3), 2018.
[85] Y. Wang, L. Wang, Y. Li, D. He, and T. Liu. A theoretical analysis of NDCG type ranking measures. In *COLT*, 2013.
[86] C. Wei, B. Wu, S. Wang, R. Lou, C. Zhan, F. Li, and Y. Cai. Analyticdb-v: A hybrid analytical engine towards query fusion for structured and unstructured data. *Proc. VLDB Endow.*, 2020.
[87] S. Wu and M. Dredze. Do explicit alignments robustly improve multilingual encoders? In B. Webber, T. Cohn, Y. He, and Y. Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 4471–4482. Association for Computational Linguistics, 2020.
[88] Y. Wu, E. Winston, D. Kaushik, and Z. C. Lipton. Domain adaptation with asymmetrically-relaxed distribution alignment. In K. Chaudhuri and R. Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, volume 97 of *Proceedings of Machine Learning Research*, pages 6872–6881. PMLR.
[89] Z. Xu, S. Wen, J. Wang, G. Liu, L. Wang, Z. Yang, L. Ding, Y. Zhang, D. Zhang, J. Xu, and B. Zheng. AMCAD: adaptive mixed-curvature representation based advertisement retrieval system. In *ICDE*, pages 3439–3452. IEEE, 2022.
[90] L. Zhang, X. Zhou, Z. Zeng, and Z. Shen. Are ID embeddings necessary? whitening pre-trained text embeddings for effective sequential recommendation. In *ICDE*, pages 530–543. IEEE, 2024.
[91] Z. Zhang, C. Jin, L. Tang, X. Liu, and X. Jin. Fast, approximate vector queries on very large unstructured datasets. In *NSDI*. USENIX Association, 2023.
[92] J. Zheng, M. Liang, Y. Yu, Y. Li, and Z. Xue. Knowledge graph enhanced multimodal transformer for image-text retrieval. In *ICDE*, pages 70–82. IEEE, 2024.