

MUST: An Effective and Scalable Framework for Multimodal Search of Target Modality

Mengzhao Wang¹, Xiangyu Ke¹, Xiaoliang Xu², Lu Chen¹, Yunjun Gao¹, Pinpin Huang², Runkai Zhu²

¹Zhejiang University, Hangzhou, China ²Hangzhou Dianzi University, Hangzhou, China

{wmzssy,xiangyu.ke,luchen,gaojy}@zju.edu.cn; {xxl,hpp,runkai.zhu}@hdu.edu.cn

Abstract—We investigate the problem of multimodal search of target modality, where the task involves enhancing a query in a specific target modality by integrating information from auxiliary modalities. The goal is to retrieve relevant objects whose contents in the target modality match the specified multimodal query. The paper first introduces two baseline approaches that integrate techniques from the Database, Information Retrieval, and Computer Vision communities. These baselines either merge the results of separate vector searches for each modality or perform a single-channel vector search by fusing all modalities. However, both baselines have limitations in terms of efficiency and accuracy as they fail to adequately consider the varying importance of fusing information across modalities. To overcome these limitations, the paper proposes a novel framework, **Multimodal Search of Target Modality**, called MUST. Our framework employs a hybrid fusion mechanism, combining different modalities at multiple stages. Notably, we leverage vector weight learning to determine the importance of each modality, thereby enhancing the accuracy of joint similarity measurement. Additionally, the proposed framework utilizes a fused proximity graph index, enabling efficient joint search for multimodal queries. MUST offers several other advantageous properties, including a pluggable design to integrate any advanced embedding techniques, user flexibility to customize weight preferences, and modularized index construction. Extensive experiments on real-world datasets demonstrate the superiority of MUST over the baselines in terms of both search accuracy and efficiency. Our framework achieves over $10\times$ faster search times while attaining an average of 93% higher accuracy. Furthermore, MUST exhibits scalability to datasets containing more than 10 million data elements.

Index Terms—multimodal search, high-dimensional vector, weight learning, proximity graph

I. INTRODUCTION

Multimodal search [1], [2], [3], [4] represents a cutting-edge approach to information retrieval that revolutionizes how we interact with vast and diverse data sources. Traditional search engines have relied predominantly on textual queries to deliver results [5]. However, with the proliferation of even expressive multimedia contents, such as images, videos, and audio [2], [6], the need for a more comprehensive search paradigm emerged [7], [8]. Multimodal search aims to address this challenge by integrating information from multiple modalities, unlocking the potential to provide richer and more contextually relevant search results, surpassing traditional search paradigms in various retrieval tasks [4], [9], [10]. By leveraging advanced techniques in natural language processing [11], computer vision [12], and data fusion [13], multimodal search systems have the capacity to revolutionize user experiences and enable applications that span industries, from e-commerce and

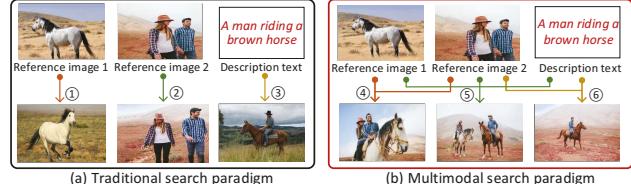


Fig. 1: An example of image search based on two different paradigms. The queries consist of two reference images and a description text. The goal is to search for images that not only resemble the input images but also modify certain aspects according to the description text.

healthcare to smart home systems and autonomous vehicles [14], [15], [16], [17], [18].

We investigate a targeted variant of multimodal search, known as **Multimodal Search of Target Modality** (MSTM), which enhances data in a specific target modality by integrating information from other modalities. In MSTM, the query input includes multiple modalities: one target modality and several auxiliary modality inputs. The target modality offers implicit context, while the auxiliary modalities introduce new traits to the target modality input. The goal is to retrieve relevant objects whose contents in the target modality align with the specified multimodal query. A key feature of MSTM is its ability to iteratively use a returned target modality example, like an image, as a reference and express differences through auxiliary modalities like text or additional images, allowing users to precisely define search criteria and obtain more tailored results based on preferences and needs.

Example 1. *In image search tasks, using a single modality as a query may not fully capture users' intentions [19]. Fig. 1(a) illustrates an image search using two reference images and a simple text description as queries. Each single modal query yields distinct outputs, highlighting the limitations of relying solely on one modality.*

In contrast, the right-hand side of Fig. 1 shows that different combinations of query inputs result in diverse outputs, each conveying a wealth of information. Additionally, when setting image 1 as the target modality in query ④, the emphasis is on the horse, while in query ⑤, the auxiliary information in the text guides the focus on the human-horse pairing while preserving all elements in the images. This exemplifies the power of utilizing MSTM queries to precisely express user preferences and obtain more comprehensive search results.

Other Applications. Apart from its direct use in image search for e-commerce, MSTM finds diverse applications across vari-

ous domains. In healthcare, MSTM enhances decision-making by augmenting medical images with patient electronic health records and symptom descriptions. This enables searching for past medical images with known decision labels, providing medical professionals with a comprehensive view of a patient’s condition and facilitating more informed decision-making. Smart home systems benefit from targeted multimodal fusion, as it allows for tailored responses to users’ voice commands, contextual information from sensors, and user profiles and configuration histories. This integration results in more personalized and efficient interactions with smart home devices. In each scenario, MSTM empowers these applications to deliver more sophisticated, relevant, and personalized outputs, significantly enhancing the user experience in the digital era.

Possible Solutions and Limitations. To tackle the MSTM problem, we propose two baselines: *Multi-streamed Retrieval* (MR) from the Database (DB) and Information Retrieval (IR) communities, and *Joint Embedding* (JE) from the Computer Vision (CV) community. Both baselines utilize advanced embedding techniques to transform multimodal inputs into high-dimensional vectors and subsequently perform vector searches to retrieve results. The main difference between the two lies in how they handle the embedding of a query, leading to distinct implementations of vector search in the context of MSTM. In MR, the query is divided into smaller subqueries, and separate solutions are applied for each modality [1]. The results from all candidate sets are merged to obtain the final query result. While this approach can use established single-modal search methods, it still suffers from accuracy and efficiency issues: The candidate sets may be too large or irrelevant due to incomplete, noisy, or ambiguous information in the target modality or auxiliary modalities [20], [21], [22]. The evaluation on million-scale data shows that it requires more than 10^4 candidates per modality to achieve the best top-100 results, yet the recall rate remains low, being less than 0.2 (Fig. 6). On the other hand, JE addresses the problem through multimodal learning. This approach embeds the features of all modality inputs into a single vector, allowing for vector search [23], [24], [25] on a corpus of vectors for the target modality. However, JE faces challenges in synergistically understanding multimodal information. The modality gap introduces ambiguity in determining what information is essential and what can be disregarded [26], making joint embedding still an open problem [3], [27]. Notably, even with the best joint embedding approach, the top-1 recall rate barely surpasses 0.4 (§VIII-B).

Our Solution. We present a novel framework for Multimodal Search of Target Modality, named MUST. This framework utilizes a hybrid fusion mechanism to combine various modalities at multiple stages, enhancing search accuracy by minimizing similarity measurement errors. Additionally, it constructs a fused proximity graph index encompassing all modal information and performs an efficient joint search. Indeed, MUST distinguishes itself from the two baselines in three main aspects. First, MUST enables the fusion of multiple modalities through a composition vector generated using multimodal learning

models like CLIP [28]. Meanwhile, MUST still supports the separate embedding of different modalities. Note that the embedding component in MUST is *pluggable*, allowing seamless integration of any newly-devised encoder into the system. Second, MUST provides a vector weight learning model to obtain the relative weights of different modalities, which projects an object to a unified high-dimensional vector space by concatenating different modal vectors with these weights. The loss function pulls the anchor closer to the positive example and pushes it away from the negative examples, based on the joint similarity in the unified space. Importantly, the learned weights capture the significance of different modalities, not their specific contents, leading to improved generalization across various query workloads. Despite the learned weights, MUST still allows users to customize their weight preferences if desired. Third, MUST builds a fused index for all modal information (not a separate index for each modality). Based on this index, it implements a joint search strategy for the multimodal query to obtain results efficiently. Notably, MUST employs a general pipeline to construct the fused index by amalgamating fined-grained components, enabling *flexibility* to seamlessly integrate these components from current proximity graphs. Furthermore, MUST enhances indexing and search performance by re-assembling index components and optimizing the multi-vector computations.

Contributions. To the best of our knowledge, this is the first work that systematically explores the MSTM problem in data embedding, importance mining, indexing, and search strategies. The main contributions are:

- We explore the MSTM problem, which enhances a query in a specific target modality by combining information from auxiliary modalities (§II). We embed objects using various encoders and build two baselines for MSTM by integrating the techniques from DB, IR, and CV communities (§III).
- We present MUST, a new framework that uses a hybrid fusion mechanism to improve search accuracy and efficiency for any modality combination of MSTM (§IV). MUST supports pluggable unimodal and multimodal embedding methods (§VIII-B) and various graph indexes (§VIII-G).
- We provide a multi-vector representation method for multimodal objects and queries (§V). Each modality of an object or query is transformed into a high-dimensional vector via unimodal or multimodal encoders. This way, we can describe an object or query more fully with multiple vectors.
- We present a lightweight and effective vector weight learning model, to get the relative weights of different modalities (§VI). The learned weights capture the importance of different modalities and adapt to various query workloads.
- We provide a component-based index construction pipeline to build a fused index for all modal information and execute a joint search of the multimodal query (§VII). Our pipeline achieves better performance by re-assembling existing components and optimizing multi-vector computations.
- We implement MUST and evaluate it on five real-world

TABLE I: Frequently used notations

Notations	Descriptions
\mathcal{S}, o	A set of multimodal objects, an object in \mathcal{S}
q	A multimodal query input
o^i, q^i	The data part of o, q in the i -th modality
m	The number of modalities in o ($o \in \mathcal{S}$)
t	The number of modalities in q ($t \leq m$, usually $t = m$)
$\phi_i(\cdot)$	The encoder for the i -th modality
$\Phi(\cdot, \dots)$	A multimodal encoder
$IP(\cdot, \cdot)$	The inner product (IP) of two vectors
SME	The similarity measure error (Eq. 4)
ω_i	The vector weight in the i -th modality
\hat{q}, \hat{o}	The concatenated vectors of q, o

datasets and four extended datasets to verify its accuracy, efficiency, and scalability (§VIII). We show that search accuracy can be improved significantly by multi-stage fusion (§VIII-B) or combining more modalities (§VIII-E).

II. PRELIMINARIES

In this section, we present the essential terminology and formally define the MSTM problem. The frequently-used notations are summarized in Table I.

Object Set. The object set \mathcal{S} consists of n objects, each $o \in \mathcal{S}$ possessing m modalities ($m \geq 1$), represented as o^i ($0 \leq i \leq m - 1$). In our focus, $m > 1$ indicates that each object in the set has multiple modalities. The versatility of an object set allows it to represent various types of data. For example, in the context of movies, each object may comprise three modalities—video, image, and text—corresponding to the movie itself, its poster, and its introduction, respectively. **Query.** A query q consists of t modalities, each represented by q^i ($0 \leq i \leq t - 1, 1 \leq t \leq m$). We focus on the case where $t > 1$ indicates a multimodal query input. In this context, we specify one of the query modalities as the target, which is used for rendering the search results¹. For simplicity, throughout the following discussion, we assume that q^0 represents the target modality. However, it is important to note that users may not always provide a multimodal query input. Our solution for solving MSTM is designed to accommodate such cases when certain modalities, including the targeted modality, might be absent. Further details are provided in §IX.

Problem Statement. Given an object set \mathcal{S} , a query input q , and a positive integer k , the goal of Multimodal Search of the Target Modality (MSTM) problem is to find k objects from \mathcal{S} that best match the query. Specifically, the target modality part of each object in the result set R should closely resemble q^0 , while also adhering to certain aspects specified by the set $\{q^i | 1 \leq i \leq t - 1\}$, which comprises the auxiliary modalities of the query. In the image search example of Fig. 1, the output image of query ⑤ contains all elements present in two reference images and also matches the provided text description. This exemplifies a successful search result that accurately captures the user’s query intent.

Performance Metric. To measure the accuracy of the search results, we use the recall rate as the evaluation metric. Suppose

¹We can also fuse other modalities into the target modality to form a composition vector, i.e., Option 2 in Fig. 4(f) and discussion in §IV.

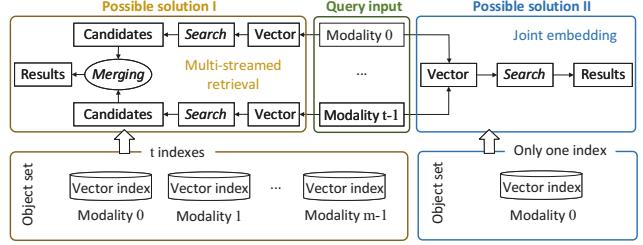


Fig. 2: Overview of two possible baselines for the MSTM problem.

there are k' ground-truth objects of q (denoted by \mathcal{G}) in the object set \mathcal{S} . The recall rate at k is formally defined as below:

$$Recall@k(k') = \frac{|R \cap \mathcal{G}|}{k'} . \quad (1)$$

III. BASELINES

To tackle the MSTM problem, we propose two baselines, namely MR and JE, leveraging current advancements.

Basic Idea. Given an object set \mathcal{S} and a query q , we transform the different modalities into high-dimensional feature vectors through embedding². These vectors capture the essence of each modality and allow for efficient comparison and retrieval [7], [13]. To solve MSTM, we conduct a vector search procedure using one or more vector indexes constructed from the feature vectors of objects in \mathcal{S} . The similarity between the query vector and potential result vectors from \mathcal{S} is evaluated using the inner product (IP). As such, we can find objects in \mathcal{S} whose target modality parts closely match the multimodal query.

Similarity Measurement Error. Unless stated otherwise, we compute the similarity between vectors using the IP metric, and all vectors are normalized. For a given object o in \mathcal{S} and a query input q , we calculate the IP between their corresponding vector representations $\phi_i(q^i)$ and $\phi_i(o^i)$ as follows:

$$IP(\phi_i(q^i), \phi_i(o^i)) = \phi_i(q^i) \odot \phi_i(o^i) , \quad (2)$$

where \odot denotes element-wise multiplication. The value of $IP(\phi_i(q^i), \phi_i(o^i)) \in [0, 1]$ indicates the similarity between o and q in the i -th modality. A higher value signifies a greater similarity. For the composition vector $\Phi(q^0, \dots, q^{t-1})$, we compute the IP w.r.t the target modality by

$$IP(\Phi(q^0, \dots, q^{t-1}), \phi_0(o^0)) = \Phi(q^0, \dots, q^{t-1}) \odot \phi_0(o^0) . \quad (3)$$

Current embedding methods can ensure that $\Phi(q^0, \dots, q^{t-1})$ and $\phi_0(o^0)$ share the same vector space [7], [28]. Eq. 3 illustrates the similarity between the query q and the target modality content of object o . The query result is the object whose target modality content exhibits the highest similarity to q . In our assumption of exact vector search, which entails no errors in the similarity computation, the query accuracy is solely dependent on the similarity measurement error (SME). For a result object r and the ground-truth result a w.r.t q , the SME is computed as follows:

$$SME(a, r) = 1 - IP(\phi_0(a^0), \phi_0(r^0)) . \quad (4)$$

²We apply state-of-the-art embedding techniques for each modality (please refer to our extended version [29] for the specific encoders used in this paper).

The *SME* reflects the encoder loss and how well the exact vector search can retrieve the ground-truth object.

Baseline 1: Multi-streamed Retrieval (MR). As depicted in Fig. 2 (upper left), MR divides the MSTM into t separate sub-queries, each focusing on a different modality. These individual sub-queries are processed independently to obtain candidate sets of potential results for each modality. To achieve this, MR uses a unimodal encoder $\phi_i(\cdot)$ to embed each query element q^i into a vector space, resulting in the feature vector $\phi_i(q^i)$ [30]. Subsequently, it builds m vector indexes on \mathcal{S} for all modalities and performs t separate vector search procedures [1]. Finally, it merges all candidates from individual sub-queries and returns the final results [20]. This framework is a common practice in research related to DB and IR [31], [21], [32], and it efficiently handles hybrid queries by effectively merging multiple constraints *with known importance*, making it a possible baseline to address MSTM.

In hybrid queries for vector similarity search with attribute constraints [31], [21], the attribute holds higher importance than the feature vector. This allows for straightforward candidate merging by identifying objects that (1) match the attribute of the query and (2) are more similar to the feature vector of the query. However, in MSTM, the importance of each modality is unknown, making it challenging to directly merge candidates based on their importance. We take the intersection of all candidates as the final results in MSTM, and further optimize this framework by replacing $\phi_0(q^0)$ with $\Phi(q^0, \dots, q^{t-1})$ obtained from the joint embedding.

Baseline 2: Joint Embedding (JE). JE leverages the advancements in multimodal representation learning [7], [13] to address the MSTM problem. It processes the multimodal query by fusing the target modality and auxiliary modality inputs into a single query vector. In Fig. 2 (upper right), both the target modality and auxiliary modality inputs are jointly embedded to create a unified vector representation $\Phi(q^0, \dots, q^{t-1})$. Once the composition vector is obtained, vector search is performed on the vector index constructed from the target modality vectors $\{\phi_0(o^0) | o \in \mathcal{S}\}$. Recently, the CV community has extensively explored multimodal encoders, leading to the design of various joint embedding networks that aim to enhance the quality of embeddings. For instance, TIRG (Text-Image Residual Gating) [7] employs a gating-residual mechanism to fuse multiple modalities effectively. Another notable work [28] introduces a combiner network that combines features from multiple modalities, derived from the OpenAI CLIP network [13]. However, these existing multimodal encoders also fail to capture the importance of different modalities, which is crucial in solving MSTM.

Summary. Both baselines employ vector search to efficiently retrieve query results, as depicted in Fig. 2. However, they differ in how they process the embedding of the query q , resulting in distinct implementations of the vector search procedure in the context of MSTM. For detailed explanations of the high-dimensional vector search, please refer to Appendix C [29].

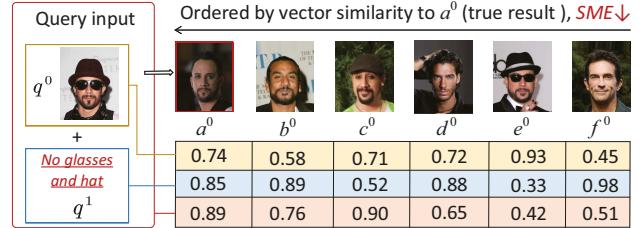


Fig. 3: A face retrieval example on CelebA with image and text modalities [33]. $a-f$ are the returned objects by different methods, only the target modality is shown. The table shows the IP between different query vectors and the upper face vectors. Three query vectors are $\phi_0(q^0)$, $\phi_1(q^1)$, and $\Phi(q^0, q^1)$, corresponding to the yellow, blue, and red rows, respectively.

IV. PROPOSED MUST FRAMEWORK: AN OVERVIEW

In the following, we discuss the limitations of the two baselines, MR and JE, in addressing MSTM. These baselines, while effective in other contexts, face challenges when it comes to handling the importance of different modalities in the multimodal query. We will highlight their shortcomings and propose a new framework, **Multimodal Search of Target Modality (MUST)**, that overcomes these issues. Our new approach aims to adapt to the varying significance of different modalities, enabling more accurate and context-aware multimodal search results in MSTM.

Example 2. In Fig. 3, we illustrate an example on a real-world face dataset, CelebA [33]. The query input q consists of a reference face q^0 and a textual constraint q^1 . The image a^0 is the ground-truth face that perfectly matches q , and b^0-f^0 are other candidate images³. These images are displayed in SME-descending order from right to left, indicating the increasing similarity with a^0 . Additionally, the table in Fig. 3 presents the IP values between different query vectors and the face vectors. For example, the first column of the table provides the IP values between $\phi_0(q^0)$ and $\phi_0(a^0)$, $\phi_1(q^1)$ and $\phi_1(a^1)$, and $\Phi(q^0, q^1)$ and $\phi_0(a^0)$, from top to bottom.

The two baselines, MR and JE, differ primarily in how they encode and utilize the multimodal query q and the object set \mathcal{S} —employing early or late fusion, respectively. In MR, q^0 and q^1 are separately encoded. Fig. 3 shows that it returns two top-1 faces, e and f , by searching for the most similar vectors concerning image and text vectors, respectively. However, these images are dissimilar to the ground-truth a^0 due to incomplete query intent. Even considering the intersection of the top-3 candidate sets $\{e, a, d\}$ for images and $\{f, b, d\}$ for text, it still returns image d instead of a , indicating the limitation of late fusion. For JE, q^0 and q^1 are combined into a composition vector $\Phi(q^0, q^1)$. By searching for the most similar vector to $\Phi(q^0, q^1)$ on $\{o^0 | o \in \mathcal{S}\}$, it erroneously returns the face c , despite using the most advanced joint embedding technique [28]. Even trying to obtain two top-3 candidate sets $\{c, a, b\}$ and $\{f, b, d\}$ by searching for the closest vectors to $\Phi(q^0, q^1)$ on $\{o^0 | o \in \mathcal{S}\}$ and $\phi_1(q^1)$ on $\{o^1 | o \in \mathcal{S}\}$, respectively, and

³We use the ResNet [12] to encode these images and calculate their SME w.r.t. a^0 . We also transform the textual constraint and image-text pair into vectors using the Encoding [34] and CLIP [28], respectively.

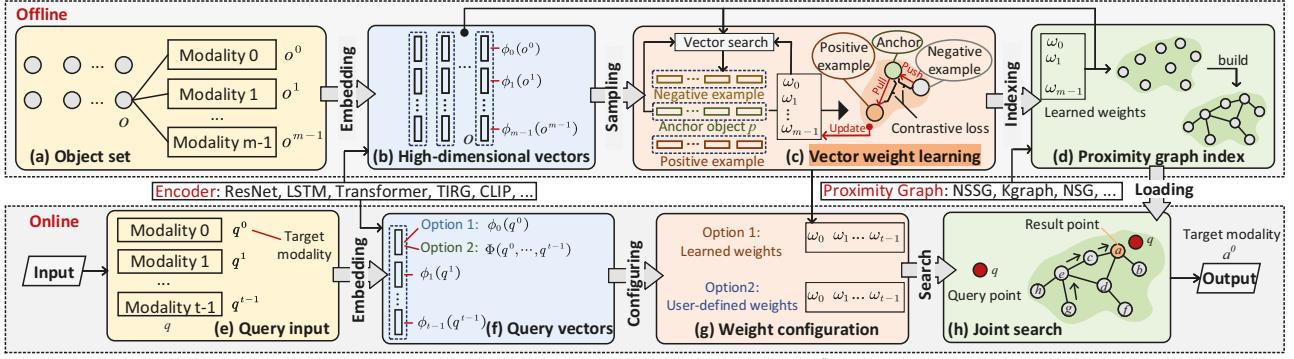


Fig. 4: Overview of the working flow of MUST.

then merging them leads to suboptimal results (please refer to §VIII-B for evaluation). This approach combines different fusion stages from the two baselines but still fails to account for the importance of different modalities, limiting accuracy and efficiency due to the merging operation. This motivates us to explore more sophisticated and comprehensive ways to fuse multiple modalities, considering the varying importance of different modalities in the multimodal query.

MUST is a novel framework tailored for addressing the MSTM, offering high accuracy, efficiency, and scalability. Unlike the two baselines discussed earlier, MUST adopts a more comprehensive approach by incorporating three *pluggable* components that *fuse multiple modalities at different levels*. This allows MUST to leverage all available modality information, taking advantage of the complementary benefits offered by different fusion levels. By doing so, MUST accurately captures the importance of different modalities while efficiently executing joint search operations on a fused index. Fig. 4 provides a high-level overview of the MUST framework, showcasing its key components, as elaborated below:

Embedding. As depicted in Fig. 4 (left), the MUST framework offers remarkable flexibility in representing objects or queries using multiple high-dimensional vectors obtained from various unimodal or multimodal encoders. It can seamlessly accommodate any encoder for any combination of modalities, such as using LSTM [35] for text, ResNet [12] for images, CLIP [28] for text-image pairs, and more. For an object set \mathcal{S} , MUST transforms each object o into m vectors from m different modalities, and each query q into t query vectors. Notably, MUST allows for flexible encoding of the target modality input. It can be encoded independently (Option 1 in Fig. 4(f)), or fused with other modalities using a multimodal encoder (Option 2 in Fig. 4(f)). By default, $\Phi(q^0, \dots, q^{t-1})$ is represented in the same vector space as $\phi_0(q^0)$ [7], ensuring compatibility and coherence within the framework. This adaptability empowers MUST to effectively handle diverse multimodal scenarios, making it a powerful and versatile solution for addressing the MSTM.

Vector Weight Learning. Innovatively, MUST introduces a vector weight learning model that discerns the importance of different modalities for similarity measurement between objects. Considering a pair of objects p and o , MUST assigns

specific weights to the vector spaces of each modality, effectively adjusting the influence of each vector. As illustrated in Fig. 4(c), MUST incorporates the weight ω_i into $\phi_i(p^i)$ to create a virtual anchor (colored green). This virtual anchor is represented by a concatenated vector $\hat{p} = [\omega_0 \cdot \phi_0(p^0), \dots, \omega_{m-1} \cdot \phi_{m-1}(p^{m-1})]$. Similarly, MUST generates the virtual point for object o by $\hat{o} = [\omega_0 \cdot \phi_0(o^0), \dots, \omega_{m-1} \cdot \phi_{m-1}(o^{m-1})]$. The joint similarity between p and o is then computed by the IP between \hat{p} and \hat{o} . To achieve the learning of vector weights, MUST utilizes vector search to identify negative examples that share a high joint similarity with p . By employing a contrastive loss function, MUST moves the virtual anchor away from the virtual points of negative examples and closer to the virtual point of the positive example. Through this process, the weights are adaptively adjusted to reflect the relative importance of different modalities in the similarity measurement. Ultimately, MUST outputs the learned weights, which can be effectively used for indexing and search operations

Indexing and Searching. MUST constructs a fused proximity graph index based on the joint similarity between objects in the object set \mathcal{S} . The weights of different modalities, acquired from the model shown in Fig. 4(c), are utilized in this process. For a query input q with t query vectors, MUST employs a merging-free joint search procedure to find the ground-truth object on the fused index. In the fused index $G = (V, E)$ (Fig. 4(d)), the objects in \mathcal{S} correspond to vertices in V , and edges in E represent similar object pairs in terms of their joint similarity. When processing a query input q , MUST’s search procedure initiates from either a random or fixed vertex (e.g., g in Fig. 4(h)) and explores neighboring vertices in G that are closer to q (e.g., e, c, a). The procedure continues until it reaches a vertex that has no neighbors closer to q than itself (e.g., a). Throughout this procedure, MUST calculates the distance of vertices from q using joint similarity. Regarding the weight options, MUST provides two choices: (1) learned weights obtained from the offline model (Option 1 in Fig. 4(g)), and (2) user-defined weights (Option 2 in Fig. 4(g)). This flexibility allows users to either leverage weights learned from the vector weight learning model or manually specify their weights for a more customized search experience.

Example 3. In the face retrieval example shown in Fig. 3, our vector weight learning model outputs the weights $\omega_0 =$

0.80 and $\omega_1 = 0.33$ for the two modalities. Leveraging these learned weights, we compute the joint similarity between the query q and the candidate objects. The concatenated vector representation of q is computed as $\hat{q} = [\omega_0 \cdot \Phi(q^0, q^1), \omega_1 \cdot \phi_1(q^1)]$. By using this joint similarity computation, we find that object a has the highest joint similarity to the query q compared to the other candidates. According to Lemma 1, we calculate $IP(\hat{q}, \hat{a})=0.6622$. As a result, MUST achieves a significantly improved query result by effectively capturing the importance of different modalities and accurately evaluating the joint similarity between objects.

V. EMBEDDING

Deep representation learning has revolutionized the use of various encoders to transform information into high-dimensional vectors, benefiting different downstream tasks [31]. Traditional encoders represent objects using single vectors from individual modalities. For example, ResNet [12] encodes face images into vectors. However, recent progress in multimodal learning has introduced encoders that can fuse multiple modalities, such as the CLIP model, which can embed both face and text as a unified vector [3]. Despite these advancements, research indicates that a single-vector representation may be inadequate for unimodal encoders, capturing only partial object information [31], and may introduce significant encoder errors for multimodal encoders [27]. Our experiments confirm that relying on a single-vector representation leads to notably low search accuracy (see Tab. III–VI).

In MUST, we introduce a novel multi-vector representation method for multimodal objects and queries (refer to Fig. 4(b) and (f)). This approach generates distinct vector representations for different modalities of an object or query. Importantly, MUST offers flexibility in encoding the target modality input. It can either be independently encoded (Option 1 in Fig. 4(f)) or fused with other modalities using a multimodal encoder (Option 2 in Fig. 4(f)). By default, $\Phi(q^0, \dots, q^{t-1})$ is represented in the same vector space as $\phi_0(q^0)$ [7], ensuring compatibility and coherence within the framework.

This approach enables us to describe an object or query comprehensively using multiple vectors, resulting in strong generalization capabilities for MUST. In scenarios where multimodal query input is unavailable, users can still perform conventional single-modal search initially and then achieve improved results through MSTM. Additionally, the embedding component in MUST is pluggable, allowing seamless integration of any newly-devised encoders into the system. Further details about the encoders are provided in Appendix B of [29].

VI. VECTOR WEIGHT LEARNING

In MUST, we combine all vectors of an object using a set of weights to form a concatenated vector. These weights serve as indicators of the importance of different modalities in representing the object. By doing so, we achieve the mapping of each object into a unified high-dimensional vector space, facilitating similarity computation between object pairs through the Inner Product (IP) of their concatenated vectors.

To determine these weights, we introduce a lightweight vector weight learning model based on *contrastive learning*. To begin, given an anchor object, we acquire its positive and negative examples. Subsequently, we construct a contrastive loss function and minimize it to learn the relative weights. The training pipeline of the model is depicted in Fig. 4(c).

A. Positive and Negative Examples

The training data consists of two parts: the anchor set Q (i.e., queries) and a set of their true resultant objects T . For each anchor $p \in Q$, there is a corresponding true object in T . Positive and negative examples for p are created as follows.

Positive Example. In T , the true object corresponding to the anchor p is directly assigned as a positive example p^+ .

Negative Examples. We focus on identifying hard negative examples that are easily confused with the true object of p . Using a weight combination $\omega_0, \omega_1, \dots, \omega_{m-1}$, we map p and objects in T into a unified vector space (shadow region in Fig. 4(c)). In this space, we generate virtual anchor and object points based on their concatenated vectors. Then, through vector search, we obtain the top- k result objects denoted by a set R with the highest similarity to p . R is defined as follows:

$$R = \arg \max_{R \subset T \wedge |R|=k} \sum_{r \in R} IP(\hat{r}, \hat{p}) , \quad (5)$$

where $\hat{r} = [\omega_0 \cdot \phi_0(r^0), \dots, \omega_{m-1} \cdot \phi_{m-1}(r^{m-1})]$ and $\hat{p} = [\omega_0 \cdot \phi_0(p^0), \dots, \omega_{m-1} \cdot \phi_{m-1}(p^{m-1})]$ are the concatenated vectors of r and p , respectively. We designate false objects in R as negative examples, denoted by $N^- = R \setminus \{p^+\}$.

B. Loss Function

Our training objective is to push the virtual anchor away from the virtual points of objects in N^- and pull it closer to the virtual point of the object p^+ . To achieve this, we devise a loss function L based on the well-known contrastive loss [36]. Let Q be a training minibatch of M anchors, and we define the loss function as follows:

$$L = \frac{1}{M} \sum_{p \in Q} -\log \frac{e^{IP(\hat{p}, \hat{p}^+)}}{e^{IP(\hat{p}, \hat{p}^+)} + \sum_{p^- \in N^-} e^{IP(\hat{p}, \hat{p}^-)}} . \quad (6)$$

We aim to minimize L to learn the relative weights, starting with a random initialization of weights $\omega_0, \omega_1, \dots, \omega_{m-1}$. These weights are used to compute concatenated vectors of an anchor p and its positive and negative examples, with negative examples obtained through vector search under the current weights. The top- k result objects R are obtained using Eq. 5. If the positive example p^+ is not in R , and for all $p^- \in N^-$, it holds that $IP(\hat{p}, \hat{p}^-) > IP(\hat{p}, \hat{p}^+)$, the loss L is significant. To minimize this loss, we update the weights using gradient descent in a way that increases $e^{IP(\hat{p}, \hat{p}^+)}$ while decreasing $\sum_{p^- \in N^-} e^{IP(\hat{p}, \hat{p}^-)}$. This weight update encourages the positive example to have a higher IP w.r.t. p while pushing the negatives to have a lower IP . Subsequently, we can obtain new negatives using the updated weights and continue the weight optimization process. We eventually arrive at a set of learned weights, under which the

true object is more likely to be retrieved as the top result in the search process. We have the following lemma:

Lemma 1. *The joint similarity of an object pair is the weighted sum of the similarity of each modality.*

Proof. For two objects a and b , their concatenated vectors are \hat{a} and \hat{b} . The IP between \hat{a} and \hat{b} can be computed by

$$IP(\hat{a}, \hat{b}) = \hat{a} \odot \hat{b} = \sum_{i=0}^{m-1} \omega_i^2 \cdot IP(\phi_i(a^i), \phi_i(b^i)) , \quad (7)$$

where $IP(\phi_i(a^i), \phi_i(b^i))$ indicates the similarity between a and b in the i -th modality. \square

The weight learning process described above plays a crucial role in capturing the significance of different modalities for representing objects. By learning the relative weights, we can effectively incorporate information from multiple modalities into the similarity computation. The search process then benefits from a holistic view of object representations, considering the diverse user intentions captured by different modalities. In the face retrieval case (Example 3) and our experimental studies (e.g., Fig. 5), this more comprehensive representation of objects leads to more meaningful and precise search results.

C. Generalization Analysis of Weight

The weight-learning component in our approach eliminates the need for specific weights for each query input. This is achieved by *learning query-independent weights that are associated with the modalities themselves*, rather than the specific content within each modality. As a result, we can employ a fixed set of weights to compute the joint similarity between any query and object in the dataset. Consider two extreme query cases on a dataset containing image and text modalities. In Case 1, the text describes what is already present in the given image, while in Case 2, the text describes something not depicted in the given image. In both scenarios, our system, MUST, consistently embeds the image with the text semantics using Option 2 in Fig. 4(f), while also separately embedding the text semantics using an unimodal encoder. For any object o , MUST computes the joint similarity between o and both types of queries using the same weights (refer to the *Learned Weights* section in §VIII-F). The similarity value is determined by the inner product (IP) between the modalities, as stated in Lemma 1. Indeed, capturing the differences between image and text contents can be effectively achieved using specific vectors rather than weights, allowing us to represent the unique characteristics of each modality while avoiding the impracticality of assigning individual weights to each object in large-scale scenarios. By employing weights to capture the importance of different modalities instead of contents, we achieve better generalization across various query workloads.

In MUST, users have the option to use custom weights for specific purposes, such as giving more weight to the text modality. In this case, the learned weights can be replaced by user-defined weights, which would prioritize objects that are more similar to the emphasized modality. The evaluation of

this option is provided in Tab. IX. Note that assigning proper weights manually can be challenging in practice. Based on our experiments (§ VIII-G), we observe that different weights significantly affect the recall rate of MSTM (Fig. 9).

VII. INDEXING AND SEARCHING

To address the efficiency and scalability challenges associated with enumerating potential objects, MUST adopts an approximate method that balances accuracy and efficiency. This involves **constructing a fused index based on the similarity of concatenated vectors**. Specifically, we utilize a proximity graph index [23], which is a sota method in the vector search domain⁴. In the fused index, $G = (V, E)$, each vertex $v \in V$ represents an object $v \in \mathcal{S}$ ⁵, and each edge $(v, u) \in E$ captures a closely related object pair (v, u) via joint similarity. The index can reduce the search space and navigate us to a true object by visiting only a few objects in \mathcal{S} , leading to better efficiency. We further improve indexing and search performance by re-assembling index components and optimizing the multi-vector computations, respectively. This ensures that our system remains efficient in handling large-scale datasets.

A. Index Construction

We present a general pipeline (Algorithm 1) for constructing fine-grained proximity graphs on CGraph⁶. The pipeline is composed of five flexible components (①–⑤). By decomposing any current proximity graph [23] into these components, we can seamlessly integrate them into our pipeline⁷. Furthermore, to enhance the capabilities of our pipeline, we amalgamate components from several state-of-the-art algorithms in the context of concatenated vectors, culminating in the creation of a new indexing algorithm.

① Initialization. This component is responsible for generating the initial neighbors for each object in \mathcal{S} . We start with randomly selecting a set of objects as neighbors $N(o)$ for any given object $o \in \mathcal{S}$ (Lines 2-3). The neighbor set $N(o)$ is then updated iteratively by visiting the neighbors $N(v)$ of each object v in $N(o)$ (Lines 4-8). During the update process, for each object u in $N(v)$ where $u \notin N(o)$, we find the object z that minimizes $IP(\hat{o}, \hat{z})$. If $IP(\hat{o}, \hat{u}) \geq IP(\hat{o}, \hat{z})$, we replace z with u in $N(o)$. This iterative process is performed for all objects in \mathcal{S} to create a high-quality initial graph. The evaluation conducted indicates that only three iterations are sufficient to achieve a graph quality of over 90% (for detailed evaluation, refer to Appendix H in our extended version [29]).

② Candidate Acquisition. This component obtains some candidate neighbors $C(o)$ for each vertex o in V from the initial graph. These candidates will serve as the potential final neighbors. For each vertex o in V , we get $C(o)$ by combining o 's initial neighbors and their neighbors (Lines 9-10).

⁴Please refer to Appendix D in our extended version [29] for detailed related work discussion about vector search.

⁵We use the same symbol for an object and its corresponding vertex.

⁶CGraph refers to the Directed Acyclic Graph framework [37].

⁷In our evaluations, we implemented some representative proximity graph algorithms, which are detailed in §VIII-G.

Algorithm 1: CONSTRUCT FUSED INDEX

Input: Object set \mathcal{S} , maximum number of neighbors γ , maximum iterations ε

Output: Fused Index $G = (V, E)$ and seed vertex g

```

1  $V \leftarrow \mathcal{S}; E \leftarrow \emptyset$                                 /* ① */
2 forall  $o \in V$  do                                         /* ② */
3    $N(o) \leftarrow \gamma$  random objects     $\triangleright$  Neighbor set
4 while iterations  $\leq \varepsilon$  do                                $\triangleright$  NNDescent
5   forall  $o \in V$  and  $v \in N(o)$  and  $u \in N(v) \setminus N(o)$  do
6      $z \leftarrow \arg \min_{z \in N(o)} IP(\hat{o}, \hat{z})$ 
7     if  $IP(\hat{o}, \hat{u}) > IP(\hat{o}, \hat{z})$  then
8        $N(o) \leftarrow N(o) \setminus \{z\} \cup \{u\}$ 
9 forall  $o \in V$  and  $v \in N(o)$  do                         /* ③ */
10   $C(o) \leftarrow N(o) \cup N(v)$   $\triangleright$  Candidate neighbor
11 forall  $o \in V$  do                                         /* ④ */
12    $v \leftarrow \arg \max_{v \in C(o)} IP(\hat{o}, \hat{v}); N(o) \leftarrow N(o) \cup \{v\}$ 
13   while  $C(o) \neq \emptyset$  and  $|N(o)| < \gamma$  do
14      $v \leftarrow \arg \max_{v \in C(o)} IP(\hat{o}, \hat{v}); C(o) \leftarrow C(o) \setminus \{v\}$ 
15     forall  $u \in N(o)$  do  $\triangleright$  MRNG strategy [25]
16       if  $IP(\hat{o}, \hat{v}) > IP(\hat{u}, \hat{v})$  then
17          $N(o) \leftarrow N(o) \cup \{v\}$ 
18  $g \leftarrow$  nearest vertex to  $\frac{1}{|V|} \sum_{o \in V} \hat{o}$           /* ⑤ */
19 Ensure connectivity by BFS from  $g$                           /* ⑥ */
20 return  $G = (V, E)$  and  $g$                                       $\triangleright E = \bigcup_{o \in V} N(o)$ 

```

③ **Neighbor Selection.** In this component, we apply a filtering process to the candidate neighbors $C(o)$ and carefully select the final neighbors $N(o)$ for each vertex o in V . The primary objective is to diversify the distribution of neighbors, which is essential for ensuring search efficiency. To achieve this, we employ the MRNG strategy [25] (Lines 11-17). For each vertex o in V , we first clear its set of final neighbors $N(o)$, then extract the vertex v that is closest to o from the candidate set $C(o)$ and include it in $N(o)$. Subsequently, we iteratively select vertices from $C(o)$ that are closest to o and satisfy the condition $IP(\hat{o}, \hat{v}) > IP(\hat{u}, \hat{v})$ for all u in the current set $N(o)$. If this condition is met for a vertex v , we add it to $N(o)$. This selection process ensures a diversified distribution of neighbors (as demonstrated in **Lemma 2**). Notably, this approach has been widely acknowledged in the literature [24], [25], [38], and our experiments further corroborate its effectiveness in the context of MSTM (§VIII-D). The parameter γ is carefully evaluated, and additional details regarding its assessment can be found in Appendix H of our [29].

Lemma 2. For any two neighbors u and v in $N(o)$, the angle $\angle uov$ (denoted by $\theta(u, o, v)$) is at least 60° .

Proof. (Sketch.) Assuming $\theta(u, o, v) < 60^\circ$ for two neighbors u and v in $N(o)$, we find that the sum of the angles $\theta(o, v, u)$ and $\theta(o, u, v)$ exceeds 120° in the triangle $\triangle uov$. Here, the inner product (IP) of two vertices is used to measure the side length between them, and smaller IP values imply longer

Algorithm 2: JOINT SEARCH

Input: Fused index $G = (V, E)$, multimodal query q , seed vertex g , number of results k , result set size $l (> k)$

Output: approximate top- k results of q

```

1  $R \leftarrow \{g\}; H \leftarrow \emptyset$ 
2  $C \leftarrow l - 1$  random vertices
3  $R \leftarrow R \cup C$                                  $\triangleright$  sorted by  $IP$  to  $q$ 
4 while  $(R \setminus H) \neq \emptyset$  do  $\triangleright$  unvisited vertices
5    $v \leftarrow$  unvisited nearest vertex to  $q$  in  $R$      $\triangleright v \notin H$ 
6    $H \leftarrow H \cup \{v\}$                                  $\triangleright$  mark  $v$  as visited
7   forall  $u \in N(v)$  and  $u \notin H$  do
8      $z \leftarrow \arg \min_{z \in R} IP(\hat{q}, \hat{z})$ 
9     if  $IP(\hat{q}, \hat{z}) < IP(\hat{q}, \hat{u})$  then
10     $R \leftarrow R \setminus \{z\} \cup \{u\}$                    $\triangleright$  update  $R$ 
11 return top- $k$  nearest vertices in  $R$ 

```

sides. By comparing the IP , either $\theta(o, v, u) > \theta(o, u, v)$ (i.e., $\theta(o, v, u) > 60^\circ$) or $\theta(o, u, v) > \theta(o, v, u)$ (i.e., $\theta(o, u, v) > 60^\circ$). **Case 1:** if $\theta(o, v, u) > \theta(o, u, v)$, it implies $IP(\hat{o}, \hat{v}) > IP(\hat{o}, \hat{u})$, resulting in vertex v being added to $N(o)$ before u . Consider the assumption, we have $IP(\hat{u}, \hat{v}) > IP(\hat{o}, \hat{u})$. Therefore, vertex u cannot be added to $N(o)$. **Case 2:** If $\theta(o, u, v) > \theta(o, v, u)$, we can swap the positions of u and v in $\triangle uov$ and arrive at the same conclusion as in Case 1. We put the detailed proof in Appendix A of our [29]. \square

④ **Seed Preprocessing.** We select a fixed seed as a start vertex for searching for different queries. We first compute the centroid of all vertices in V with their concatenated vectors. We then compute the IP between each vertex and centroid to find the vertex closest to the centroid as the seed (Line 18).

⑤ **Connectivity.** We perform a breadth-first search (BFS) from the seed. In case the BFS cannot reach all vertices in V from the seed, a connection is established between a visited vertex and an unvisited vertex. This connection bridges the gap between previously unexplored regions of the graph and the BFS is continued until all vertices are reachable from the seed (Line 19), thereby enhancing the search accuracy.

B. Joint Search

Upon receiving a multimodal query input q , MUST conducts a joint search across all modalities using the fused index. Initially, q is transformed into t query vectors (Fig. 4(f)) and concatenated with a set of weights to be a virtual query point (Fig. 4(g)). When $t = m$, q is mapped into the same vector space as the objects in \mathcal{S} , enabling the computation of the inner product (IP) between q and the objects in \mathcal{S} based on Lemma 1. However, if $t \neq m$, the concatenated vectors compute the IP by setting $\omega_i = 0$ for $t \leq i \leq m - 1$. Next, the search process begins at the seed and employs greedy routing within the fused index to obtain approximate top- k results (Fig. 4(h)).

Algorithm 2 presents the joint search procedure of MUST. During the greedy routing, two key data structures, R and H , are utilized (Line 1). R represents the result set with a fixed

size of l and is initialized with the seed vertex g and $l - 1$ randomly chosen vertices. On the other hand, H is a set that keeps track of visited vertices, effectively avoiding redundant vector computations. The iterative greedy routing (Lines 4-10) selects unvisited vertices from R closest to the query point q . It calculates the IP between each neighbor of v and q , updating R accordingly. The process continues until all vertices in R are visited, yielding the top- k nearest vertices. In practice, users have the flexibility to balance accuracy and efficiency by tuning the parameter l . The value of l determines the size of the result set R and influences the trade-off between accuracy and efficiency. We conduct evaluations of l in Appendix I in our extended version [29]. Given the number of iterations η , we have the following lemma to ensure the joint similarity is non-decreasing during searching:

Lemma 3. *The sum of the IP between the query q and the vertices in R is a monotonically non-decreasing function of η , denoted by $f(\eta)$.*

Proof. In the joint search process of MUST, let's consider any two consecutive iterations $\eta = i$ and $\eta = j$ ($i < j$). We denote R_i and R_j as the R sets after the i -th and j -th iterations, respectively. Additionally, let z be the vertex farthest from q in R_i . During the j -th iteration, we encounter two cases for any neighbor u of the currently visited vertex. Case 1: If $IP(\hat{q}, \hat{z}) \geq IP(\hat{q}, \hat{u})$, R_i remains unchanged, resulting in $f(i) = f(j)$. Case 2: If $IP(\hat{q}, \hat{z}) < IP(\hat{q}, \hat{u})$, z is replaced by u in R_j , leading to $f(j) = f(i) - IP(\hat{q}, \hat{z}) + IP(\hat{q}, \hat{u})$, which implies $f(i) < f(j)$. Thus, we can deduce that $i < j$ implies $f(i) \leq f(j)$, demonstrating that $f(\eta)$ is a monotonically non-decreasing function of η . \square

Optimizing Multi-vector Computation. In our approach, the joint search, particularly the multi-vector computation, constitutes the most time-consuming part. For each object pair, we must compute m similarities between high-dimensional vectors. It is well-documented in the literature [39], [40] that vector computation can consume up to 90% of the total search time in many real-world datasets. When processing an object u (Line 7 in Algorithm 2), we need to compute the inner product of \hat{u} and \hat{q} . The resulting inner product value is then used for the similarity comparison (Line 9 in Algorithm 2) with the most dissimilar object z in R . If u is more similar to q than z , we update R with u based on this inner product value. However, if u is less similar to q than z , we can simply discard u . In this case, there is no need to compute the exact value of $IP(\hat{q}, \hat{u})$. Since the vectors are normalized, we have

$$IP(\hat{q}, \hat{u}) = 1 - \frac{1}{2} \cdot \|\hat{q}, \hat{u}\|^2 , \quad (8)$$

where $\|\hat{q}, \hat{u}\|$ is the Euclidean distance between \hat{q} and \hat{u} . As the $\|\hat{q}, \hat{u}\|^2$ increases, $IP(\hat{q}, \hat{u})$ decreases. Therefore, we scan the vectors of \hat{u} incrementally and compute the partial square Euclidean distance based on the scanned x vectors as

$$\|\widetilde{\hat{q}, \hat{u}}\|^2 = \sum_{i=0}^{x-1} \omega_i^2 \cdot \|\phi_i(u^i), \phi_i(q^i)\|^2 , \quad (9)$$

TABLE II: Dataset statistics (* marks the target modality).

Dataset	# Modality	# Object	# Query	Type	Source
CelebA [33]	2	191,549	34,326	Image*,Text	real-world
MIT-States [41]	2	53,743	72,732	Image*,Text	real-world
Shopping [42]	2	96,009	47,658	Image*,Text	real-world
MS-COCO [43]	3	19,711	1237	Image* × 2,Text	real-world
CelebA+ [33]	4	191,549	34,326	Image* × 3,Text	real-world
ImageText1M [44]	2	1,000,000	10,000	Image*,Text	semi-synthetic
AudioText1M [45]	2	992,272	200	Audio*,Text	semi-synthetic
VideoText1M [46]	2	1,000,000	10,000	Video*,Text	semi-synthetic
ImageText16M [47]	2	16,000,000	10,000	Image*,Text	semi-synthetic

where $\|\phi_i(u^i), \phi_i(q^i)\|$ is the Euclidean distance between the vectors in the i -th modality. Then, we can compute the partial IP $IP(\hat{q}, \hat{u})$ by applying Eq. 9 to Eq. 8. We check whether $IP(\hat{q}, \hat{z}) \geq IP(\hat{q}, \hat{u})$, if it holds, we can discard u immediately, otherwise, we continue to consider the next vector until scanning all vectors (i.e., $x = m$) or $IP(\hat{q}, \hat{u})$ is not more than $IP(\hat{q}, \hat{z})$. As we will demonstrate in our experiment, this optimization significantly improves the search efficiency without incurring any accuracy loss (Lemma 4).

Lemma 4. *By utilizing the multi-vector computation optimization, we can safely discard the object u that satisfies $IP(\hat{q}, \hat{z}) \geq IP(\hat{q}, \hat{u})$ by the partial IP $IP(\hat{q}, \hat{u})$. Furthermore, when $IP(\hat{q}, \hat{z}) < IP(\hat{q}, \hat{u})$, we can obtain the exact value of $IP(\hat{q}, \hat{u})$.*

Proof. According to Eq. 8, a larger value of $\|\hat{q}, \hat{u}\|^2$ corresponds to a smaller value of $IP(\hat{q}, \hat{u})$. As we incrementally scan the vectors, $\|\widetilde{\hat{q}, \hat{u}}\|^2$ gradually increases while $IP(\hat{q}, \hat{u})$ gradually decreases. Let x be the number of scanned vectors. Once $IP(\hat{q}, \hat{z}) \geq IP(\hat{q}, \hat{u})$ is true for the first time, it remains true for larger values of x . Therefore, we can safely terminate the multi-vector computation when $IP(\hat{q}, \hat{z}) \geq IP(\hat{q}, \hat{u})$. In the case where $IP(\hat{q}, \hat{z}) < IP(\hat{q}, \hat{u})$, we have $IP(\hat{q}, \hat{z}) < IP(\hat{q}, \hat{u})$ for any value of x . Hence, in this case, we scan all m vectors and obtain the exact value of $IP(\hat{q}, \hat{u})$. \square

VIII. EXPERIMENTS

We thoroughly evaluate MUST across six key aspects: (1) accuracy, (2) case study, (3) efficiency, (4) scalability, (5) query workloads, and (6) ablation studies. Kindly refer to our GitHub repository: <https://github.com/ZJU-DAILY/MUST> for our source code, datasets, and additional evaluations.

A. Experimental Setting

Datasets. We use nine datasets obtained from public sources, each with varying modalities and cardinalities, as shown in Tab. II. Unless specified otherwise, the queries consist of the same number of modalities as the objects in each dataset (i.e., $t = m$). For more details, kindly refer to Appendix J [29].

Compared Methods. We compare our proposed MUST with two baselines: Multi-streamed Retrieval (abbr. MR) and Joint Embedding (abbr. JE). To ensure a fair comparison, we use the same encoders and proximity graph index in all competitors.

Metrics. We measure search accuracy for a batch of queries by mean recall rate ($Recall@k(k')$, Eq. 1) and mean similarity measure error (SME , Eq. 4). We use queries per second (QPS) to measure search efficiency. QPS is the number of queries ($\#q$) divided by the total response time (τ), i.e., $\#q/\tau$.

TABLE III: Search accuracy on MIT-States.

Framework	Encoder	Recall@1(1)	Recall@5(1)	Recall@10(1)	SME
JE	TIRG	0.1181	0.3027	0.4175	0.1574
	CLIP	0.2236	0.4979	0.6187	0.1382
MR	ResNet17+LSTM	0.3998	0.6336	0.7106	0.1222
	ResNet50+LSTM	0.5401	0.7104	0.7639	0.1012
	ResNet17+Transformer	0.2435	0.4110	0.4931	0.1381
	ResNet50+Transformer	0.3112	0.4475	0.5142	0.1404
	TIRG+LSTM	0.3768	0.6574	0.7691	0.1283
	TIRG+Transformer	0.2830	0.4918	0.5834	0.1395
	CLIP+LSTM	0.4911	0.7619	0.8436	0.1108
MUST (ours)	CLIP+Transformer	0.3707	0.5912	0.6751	0.1285
	ResNet17+LSTM	0.5275	0.7897	0.8780	0.0915
	ResNet50+LSTM	0.6655(↑23.2%)	0.8558(↑12.3%)	0.9127(↑8.2%)	0.0738
	ResNet17+Transformer	0.3325	0.4828	0.5548	0.1272
	ResNet50+Transformer	0.3743	0.4866	0.5367	0.1344
	TIRG+LSTM	0.4202	0.7012	0.8137	0.1184
	TIRG+Transformer	0.3131	0.4800	0.5543	0.1333
CLIP+LSTM	0.5376	0.7859	0.8678	0.1006	
	CLIP+Transformer	0.4190	0.5262	0.5731	0.1229

TABLE IV: Search accuracy on CelebA.

Framework	Encoder	Recall@1(1)	Recall@5(1)	Recall@10(1)	SME
JE	TIRG	0.2725	0.5258	0.6220	0.1896
	CLIP	0.3644	0.7006	0.7789	0.1453
MR	ResNet17+Encoding	0.3337	0.5477	0.6233	0.1724
	ResNet50+Encoding	0.3098	0.5029	0.5717	0.2047
MUST (ours)	TIRG+Encoding	0.3275	0.5707	0.6622	0.1875
	CLIP+Encoding	0.4578	0.7319	0.7990	0.1416
	ResNet17+Encoding	0.5701	0.7888	0.8446	0.1087
	ResNet50+Encoding	0.5423	0.7539	0.8106	0.1293
CLIP+Encoding	TIRG+Encoding	0.4932	0.7377	0.8099	0.1433
	CLIP+Encoding	0.6388(↑39.5%)	0.8583(↑17.3%)	0.9024(↑12.9%)	0.0952

TABLE V: Search accuracy on Shopping (T-shirt).

Framework	Encoder	Recall@1(1)	Recall@5(1)	Recall@10(1)	SME
JE	TIRG	0.1320	0.4005	0.5162	0.0964
	CLIP				
MR	ResNet17+Encoding	0.0027	0.0190	0.0399	0.1379
	TIRG+Encoding	0.1320	0.4015	0.5206	0.0964
MUST (ours)	ResNet17+Encoding	0.4208	0.6931	0.7973	0.0743
	TIRG+Encoding	0.4669(↑253.7%)	0.7585(↑88.9%)	0.8507(↑63.4%)	0.0651

TABLE VI: Search accuracy on MS-COCO.

Framework	Encoder	Recall@10(1)	Recall@50(1)	Recall@100(1)
JE	MPC	0.0202	0.0865	0.1512
MR	MPC+GRU+ResNet50	0.0647	0.1827	0.2741
	ResNet50+GRU+ResNet50	0.0493	0.1633	0.2425
MUST (ours)	MPC+GRU+ResNet50	0.0825	0.2272	0.3363
	ResNet50+GRU+ResNet50	0.0914(↑41.3%)	0.2498(↑36.7%)	0.3711(↑35.4%)

Setup and Parameters. All experiments are conducted on a Linux server equipped with an Intel(R) Xeon(R) Gold 6248R CPU running at 3.00GHz and 755G memory. We perform three repeated trials and report the average results for all evaluation metrics. Due to the space limitation, we put the detailed settings in Appendix F of our extended version [29].

B. Accuracy Evaluation

In our evaluation of all methods, we employ various encoders on four real-world datasets (cf. Appendix B [29]). For JE, we utilize multimodal encoders such as TIRG [7], CLIP [13], and MPC [43] to embed all modalities into the vector space of the target modality. In the case of MR and MUST, we employ unimodal encoders, such as ResNet [12] and Transformer [11], to individually embed each modality. Additionally, we obtain a composition vector using a multimodal encoder (such as CLIP [13]), which is then used to replace the vector representation of the target modality.

Tab. III–VI show the search accuracy and *SME* of the three frameworks. We have three major observations as summarized below: First, MUST significantly outperforms its competitors on all experimental datasets. Notably, MUST achieves at least 198% and 23% improvement over JE and MR respectively for their best *Recall@1(1)* on MIT-States. MUST also reduces

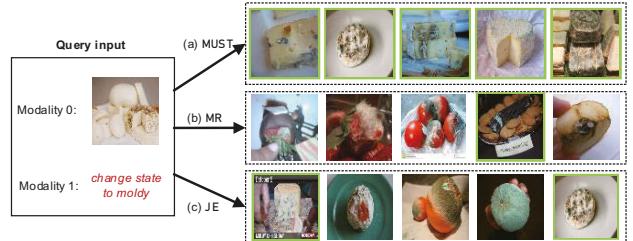


Fig. 5: Top-5 examples of different frameworks on MIT-States. The green box marks the ground-truth objects.

the *SME* on all datasets. Second, different encoders yield varying recall rates, e.g., CLIP, being a state-of-the-art multimodal encoder, achieves the highest accuracy in single-vector representation. This underscores the importance of encoder selection in achieving optimal performance. An advantage of MUST is its pluggable embedding component, which allows seamless integration of newly-devised encoders. Third, multi-vector representation exhibits higher recall rates. For example, MR (CLIP+LSTM) and MUST (CLIP+LSTM) are better than JE (CLIP) on MIT-States. Even with the same multi-vector representation, MUST consistently achieves larger improvements compared to MR. On the most challenging MS-COCO dataset, both MR and MUST demonstrate impressive performance compared to JE, which struggles due to fusing three modalities, leading to larger embedding errors.

C. Case Study

Fig. 5 shows some case studies of the MSTM problem on MIT-States. We use the best encoder for each framework based on Table III (CLIP for JE, ResNet50+LSTM for MR and MUST). We give a query input with an *image of fresh cheese* and a *text description of “change state to moldy”* and show the top-5 search results from different frameworks. The results show that MUST outperforms its competitors. The objects returned by MUST all satisfy the multimodal constraints, while most objects from MR and JE only match some of the requirements. We provide more recall examples of other queries and datasets in Appendix O [29].

D. Efficiency Evaluation

We evaluate the performance of MUST’s fused index and joint search strategy on three million-scale datasets. We conduct comparisons with MR⁸, which applies the same index and search strategy to each vector set. Additionally, we implement brute-force versions for vector search of both MUST and MR, labeled as MUST-- and MR--, respectively.

Fig. 6 presents the QPS vs Recall comparison, where we adjust the parameter *l* in Algorithm 2 to achieve different recall rates. For clarity, we exclude MUST-- and MR-- on AudioText1M and VideoText1M due to their slow performance. The results yield the following observations: First, MUST is 10× faster than MR with the same recall rate, which can be attributed to the time-consuming merging operation employed

⁸For fairness, we exclude JE as it only utilizes a single-vector representation and exhibits much lower accuracy.

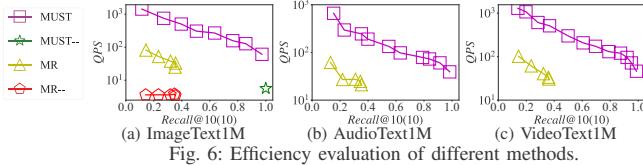


Fig. 6: Efficiency evaluation of different methods.

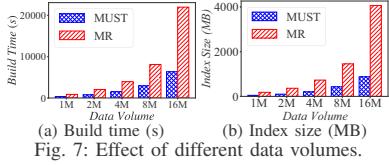


Fig. 7: Effect of different data volumes.

by MR. Second, the recall rate of MR is less than 0.4. We find that the merging operation causes the accuracy of MR to become non-increasing with increasing l . Initially, the recall rate of MR increases as l grows, owing to the increased chances of finding the target when intersecting results from each modality. However, as l further increases, the size of the intersection often exceeds k , making it challenging to identify the top- k results. Note that the importance of different modalities in MSTM is unknown, which may necessitate additional optimization for selecting the top- k objects from a large set. Third, both MUST and MR are more than 10× faster than their brute-force counterparts, which indicates the effectiveness of our indexing and searching strategies.

E. Scalability

Data Volume (n). Tab. VII shows the response time of MUST and MUST-- with varying n . We find that the response time of MUST-- increases linearly with the growth of n . In contrast, MUST exhibits only a slight increase in response time even with large n and reduces the response time by up to 98.4% when n is 16 million. Fig. 7 illustrates that MUST’s build time and index size are significantly lower than MR, affirming its efficiency and scalability in large-scale scenarios.

Number of Modalities (m). Tab. VIII reports the recall rates of MUST and MR with different m . Overall, the recall rate increases with m for both methods, as more information leads to more accurate results. However, the challenge of merging becomes more pronounced in MR as the number of modalities increases. As a result, the recall rate of $m = 4$ in MR is even lower than that of $m = 3$. This highlights MUST’s capability to effectively handle multiple modalities.

F. Query Workloads

Number of Results (k). In Fig. 8, we compare the search performance of MUST and MR with different k on ImageText1M. The results show that MUST consistently outperforms MR for any k . Moreover, MUST brings more improvements on larger k while MR has a limited recall rate and QPS (cf. §VIII-D). This is because MR requires more candidates from each modality when k is larger, which makes merging even more challenging, e.g., the number of candidates is 1,300 for the best $Recall@1(1)$, and 10,500 for the best $Recall@100(100)$.

Learned Weights. We investigate the impact of different queries with fixed learned weights on MIT-States. In Fig. 5,

TABLE VII: Response time comparison (in seconds) of MUST-- and MUST when $Recall@10(10) > 0.99$ under different data volumes. The value in parentheses shows the percentage of response time decrease by using MUST.

Scale	1M	2M	4M	8M	16M
MUST--	15.4	32.8	67.5	129.9	266.9
MUST	2.7 (↓82.5%)	2.7 (↓91.8%)	3.4 (↓95.0%)	3.4 (↓97.4%)	4.4 (↓98.4%)

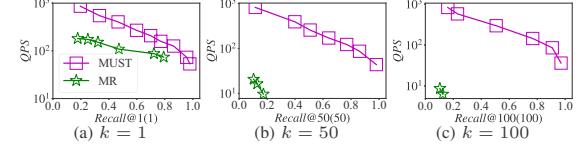


Fig. 8: Effect of different k .

the original text describes something not present in the given image. To create a new query input, we retain the reference image while modifying the text description to “remove the fresh state”. This query now describes what is already present in the given image. Both queries are executed with the same learned weights on MUST, and we observe that they yield identical query results. This compelling result verifies the generalization capability of the fixed learned weights, highlighting that the learned weights reflect the importance of different modalities, independent of their specific content.

User-defined Weights. Tab. IX shows the effect of MUST with different user-defined weights on MIT-States. We calculate the mean similarity over one modality for a batch of query inputs and returned objects. For example, when $\omega_0^2 = \omega_1^2 = 0.5$, the mean IP between modality 0 of query inputs and returned objects is 0.6915 and between modality 1 is 0.9999. To get an object whose modality 0 is more similar to the query input, users can increase the weight of modality 0. When $\omega_0^2 = 0.9$ and $\omega_1^2 = 0.1$, the returned object has higher similarity to the query input in modality 0. Thus, we can get customized results by adjusting the weight configuration.

Number of Query Modalities (t). We study how different t values in the queries affect the search accuracy on MIT-States. Tab. X shows the search accuracy when only one modality is used in the queries (i.e., $t = 1$). Compared with multimodal queries ($t = 2$, cf. Tab. III), the single-modal queries have lower search accuracy. Thus, using more query modalities is crucial for the quality of query results.

G. Ablation Study

Vector Weight Learning Model. We compare the proposed hard negative acquisition strategy and the random selection. Fig. 9 shows the loss and recall rate w.r.t. the epoch on ImageText1M. We can observe that the model using the hard negatives converges faster compared to the model using the random ones. Additionally, the learned weights from the hard negatives lead to a higher recall rate, demonstrating the effectiveness of our strategy. It is important to highlight that the weight learning model is remarkably efficient, as it takes less than 200 seconds to train on all datasets. In contrast, the embedding models used in the process require over 12 hours to train. As a result, the vector weight learning model is lightweight and imposes minimal additional training costs.

TABLE IX: Effect of different user-defined weights. q is the query input and r is the returned result.

Weights	ω_0^2	0.5	0.6	0.7	0.8	0.9
	ω_1^2	0.5	0.4	0.3	0.2	0.1
$IP(\phi_0(q^0), \phi_0(r^0))$	0.6915	0.7009	0.7440	0.8286	0.9301	
$IP(\phi_1(q^1), \phi_1(r^1))$	0.9999	0.9960	0.9748	0.9242	0.8525	

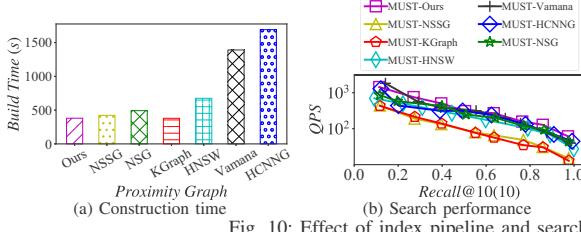


Fig. 10: Effect of index pipeline and search optimization.

Proximity Graph. We implement six proximity graphs in MUST: KGraph [48], NSG [25], NSSG [49], HNSW [24], Vamana [50], and HCNG [51]. We also re-assemble KGraph, NSG, and NSSG according to §VII-A to form our fused index. We evaluate their indexing and search performance on ImageText1M. Fig. 10(a) shows the index construction time of different methods. Our optimized one is faster than the others. Therefore, our pipeline facilitates the design of proximity graph algorithms and can improve performance even without new optimization. Fig. 11 shows the visualization of three neighbors for an object on CelebA. The vertex and its neighbors in MUST’s index balance the importance of different modalities and have a better joint similarity. The vertex and its neighbors in MR’s indexes only consider the similarity in one modality.

Multi-vector Computation Optimization. Fig. 10(c) shows the effect of multi-vector computation optimization on ImageText1M. This optimization improves the search efficiency without affecting the search accuracy. This is because we can skip some vector computations without losing accuracy by scanning the vectors of each object and query incrementally (cf. Lemma 4). This optimization is more significant in high-accuracy regions than in low-accuracy regions.

IX. DISCUSSION

Single Modality Inputs. In scenarios where users provide single-modal query inputs but seek more personalized results, MUST adapts by refining the query iteratively using a returned target modality example. For instance, in image retrieval, users may only provide text input. MUST can then generate an output image based on the given text, serving as a reference for users to enhance their query by adding additional text. This interactive process enables users to create more complete multimodal query inputs and obtain the desired results using MUST, even if the initial inputs are incomplete or imprecise.

Index Updates. The index in MUST relies on a proximity graph algorithm, and the efficacy of dynamic updates depends on the specific proximity graph employed. While certain algorithms, like KGraph [48] and NSG [25], do not support dynamic updates, others, such as HNSW [24] and Vamana

TABLE X: Effect of single query modality on MIT-States.

Modality	Encoder	Recall@1(1)	Recall@5(1)
Target	ResNet17	0.0268	0.1103
	ResNet50	0.0363	0.1393
Auxiliary	LSTM	0.2747	0.4343
	Transformer	0.2601	0.2641

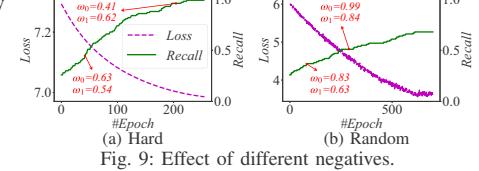


Fig. 9: Effect of different negatives.

[50], adeptly handle dynamic updates by incrementally inserting data points. For instance, upon the arrival of a new object, its embedding vector can be used to search for neighbors in the index, updating them accordingly. However, it is crucial to note that all existing proximity graph algorithms necessitate periodic reconstruction to maintain optimal performance [21]. For example, a deleted data point is not immediately removed from the index, and it can be marked with a data-status bitset. This is because the data point may be essential to ensure the connectivity of the proximity graph. The actual deletion takes place during the reconstruction process. However, this process is time-consuming for proximity graph algorithms, which affects the scalability of MUST in dynamic data update scenarios. Therefore, supporting efficient index updates in MUST remains an ongoing concern.

X. CONCLUSION

In this study, we thoroughly investigate the MST problem and propose a novel and effective framework called MUST. Our framework introduces a hybrid fusion mechanism that intelligently combines different modalities at multiple stages, capturing their relative importance and accurately measuring the joint similarity between objects. Additionally, we have developed a fused proximity graph index and an efficient joint search strategy tailored for multimodal queries. MUST exhibits the capability to handle interactive multimodal search scenarios, wherein users may lack query inputs for certain modalities. The comprehensive experimental results demonstrate the superior performance of MUST compared to the baselines, showcasing its advantages in terms of accuracy, efficiency, and scalability. For more in-depth discussions and analysis, we refer readers to Appendix N [29].

Looking ahead, we plan to further enrich MUST by incorporating additional encoders such as the OpenAI embeddings [52] and Hugging Face embeddings [53].

ACKNOWLEDGMENT

This work was supported in part by the NSFC under Grants No. (62102351, 62025206, U23A20296) and the Yongjiang Talent Programme (2022A-237-G). Lu Chen is the corresponding author of the work.

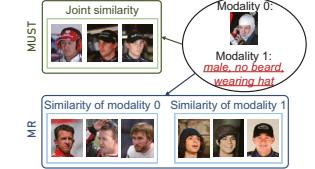


Fig. 11: An example of the top-3 neighbors on different indexes.

REFERENCES

- [1] I. Tautkute, T. Trzciński, A. P. Skorupa, Ł. Brocki, and K. Marasek, “Deepstyle: Multimodal search engine for fashion and interior design,” *IEEE Access*, vol. 7, pp. 84 613–84 628, 2019.
- [2] J. Etzold, A. Brousseau, P. Grimm, and T. Steiner, “Context-aware querying for multimodal search engines,” 2012.
- [3] S. Jandial, P. Badjatiya, P. Chawla, A. Chopra, M. Sarkar, and B. Krishnamurthy, “Sac: Semantic attention composition for text-conditioned image retrieval,” in *CVPR*, 2022, pp. 4021–4030.
- [4] H. Wen, X. Song, X. Yang, Y. Zhan, and L. Nie, “Comprehensive linguistic-visual composition network for image retrieval,” in *SIGIR*, 2021, pp. 1369–1378.
- [5] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval: The Concepts and Technology behind Search*, 2nd ed. USA: Addison-Wesley Publishing Company, 2011.
- [6] T. Baltrušaitis, C. Ahuja, and L.-P. Morency, “Multimodal machine learning: A survey and taxonomy,” *TPAMI*, vol. 41, no. 2, pp. 423–443, 2018.
- [7] N. Vo, L. Jiang, C. Sun, K. Murphy, L.-J. Li, L. Fei-Fei, and J. Hays, “Composing text and image for image retrieval—an empirical odyssey,” in *CVPR*, 2019, pp. 6439–6448.
- [8] M. Patel, T. Gokhale, C. Baral, and Y. Yang, “CRIPP-VQA: counterfactual reasoning about implicit physical properties via video question answering,” in *EMNLP*, 2022, pp. 9856–9870.
- [9] “Mum brings multimodal search to lens, deeper understanding of videos and new serp features,” <https://searchengineland.com/mum-brings-multimodal-search-to-lens-deeper-understanding-of-videos-and-new-serp-features-374798>, 2021.
- [10] T. Yu, Y. Yang, Y. Li, L. Liu, M. Sun, and P. Li, “Multi-modal dictionary bert for cross-modal video search in baidu advertising,” in *CIKM*, 2021, pp. 4341–4351.
- [11] J. Devlin, M. Chang, K. Lee, and K. Toutanova, “BERT: pre-training of deep bidirectional transformers for language understanding,” in *NAACL*, 2019, pp. 4171–4186.
- [12] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition,” in *CVPR*, 2016, pp. 770–778.
- [13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, “Learning transferable visual models from natural language supervision,” in *ICML*, 2021, pp. 8748–8763.
- [14] S. Liu, L. Li, J. Song, Y. Yang, and X. Zeng, “Multimodal pre-training with self-distillation for product understanding in e-commerce,” in *WSDM*, 2023, p. 1039–1047.
- [15] M. Levy, R. Ben-Ari, N. Darshan, and D. Lischinski, “Chatting makes perfect-chat-based image retrieval,” *arXiv:2305.20062*, 2023.
- [16] V. Ramanishka, “Describing and retrieving visual content using natural language,” Ph.D. dissertation, Boston University, 2020.
- [17] J. Ma, Y. Chen, F. Wu, X. Ji, and Y. Ding, “Multimodal reinforcement learning with effective state representation learning,” in *AAMAS*, 2022, pp. 1684–1686.
- [18] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra, “Improving vision-and-language navigation with image-text pairs from the web,” in *ECCV*, 2020, pp. 259–274.
- [19] Y. Zhao, Y. Song, and Q. Jin, “Progressive learning for image retrieval with hybrid-modality queries,” in *SIGIR*, 2022, pp. 1012–1021.
- [20] K. Zagoris, A. Arampatzis, and S. A. Chatzichristofis, “www. mmretrieval. net: a multimodal search engine,” in *SISAP*, 2010, pp. 117–118.
- [21] 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,” *PVLDB*, vol. 13, no. 12, pp. 3152–3165, 2020.
- [22] S. Zhang, M. Yang, T. Cour, K. Yu, and D. N. Metaxas, “Query specific rank fusion for image retrieval,” *TPAMI*, vol. 37, no. 4, pp. 803–815, 2014.
- [23] M. Wang, X. Xu, Q. Yue, and Y. Wang, “A comprehensive survey and experimental comparison of graph-based approximate nearest neighbor search,” *PVLDB*, vol. 14, no. 11, pp. 1964–1978, 2021.
- [24] Y. A. Malkov and D. A. Yashunin, “Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs,” *TPAMI*, vol. 42, no. 4, pp. 824–836, 2020.
- [25] C. Fu, C. Xiang, C. Wang, and D. Cai, “Fast approximate nearest neighbor search with the navigating spreading-out graph,” *PVLDB*, vol. 12, no. 5, pp. 461–474, 2019.
- [26] F. Zhang, M. Xu, Q. Mao, and C. Xu, “Joint attribute manipulation and modality alignment learning for composing text and image to image retrieval,” in *ACM MM*, 2020, pp. 3367–3376.
- [27] G. Delmas, R. S. de Rezende, G. Csurka, and D. Larlus, “ARTEMIS: attention-based retrieval with text-explicit matching and implicit similarity,” in *ICLR*, 2022.
- [28] A. Baldari, M. Bertini, T. Uricchio, and A. Del Bimbo, “Conditioned and composed image retrieval combining and partially fine-tuning clip-based features,” in *CVPR*, 2022, pp. 4959–4968.
- [29] M. Wang, X. Ke, X. Xu, L. Chen, Y. Gao, P. Huang, and R. Zhu, “MUST: An effective and scalable framework for multimodal search of target modality,” *arXiv:2312.06397*, 2023.
- [30] L. Kennedy, S.-F. Chang, and A. Natsev, “Query-adaptive fusion for multimodal search,” *Proceedings of the IEEE*, vol. 96, no. 4, pp. 567–588, 2008.
- [31] 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, pp. 2614–2627.
- [32] M. Wang, L. Lv, X. Xu, Y. Wang, Q. Yue, and J. Ni, “Navigable proximity graph-driven native hybrid queries with structured and unstructured constraints,” *arXiv:2203.13601*, 2022.
- [33] Z. Liu, P. Luo, X. Wang, and X. Tang, “Deep learning face attributes in the wild,” in *ICCV*, 2015, pp. 3730–3738.
- [34] Z. Wang, B. Fan, G. Wang, and F. Wu, “Exploring local and overall ordinal information for robust feature description,” *TPAMI*, vol. 38, no. 11, pp. 2198–2211, 2015.
- [35] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, “Lstm: A search space odyssey,” *TNNLS*, vol. 28, no. 10, pp. 2222–2232, 2016.
- [36] J. D. Robinson, C. Chuang, S. Sra, and S. Jegelka, “Contrastive learning with hard negative samples,” in *ICLR*, 2021.
- [37] ChunelFeng, “CGraph,” <https://github.com/ChunelFeng/CGraph>, 2021.
- [38] W. Li, Y. Zhang, Y. Sun, W. Wang, M. Li, W. Zhang, and X. Lin, “Approximate nearest neighbor search on high dimensional data - experiments, analyses, and improvement,” *TKDE*, vol. 32, no. 8, pp. 1475–1488, 2020.
- [39] W. Zhao, S. Tan, and P. Li, “SONG: approximate nearest neighbor search on GPU,” in *ICDE*, 2020, pp. 1033–1044.
- [40] J. Gao and C. Long, “High-dimensional approximate nearest neighbor search: with reliable and efficient distance comparison operations,” *SIGMOD*, vol. 1, no. 2, pp. 137:1–137:27, 2023.
- [41] P. Isola, J. J. Lim, and E. H. Adelson, “Discovering states and transformations in image collections,” in *CVPR*, 2015, pp. 1383–1391.
- [42] Y. Hou, E. Vig, M. Donoser, and L. Bazzani, “Learning attribute-driven disentangled representations for interactive fashion retrieval,” in *ICCV*, 2021.
- [43] A. Neculai, Y. Chen, and Z. Akata, “Probabilistic compositional embeddings for multimodal image retrieval,” in *CVPR*, 2022, pp. 4547–4557.
- [44] “Datasets for approximate nearest neighbor search,” <http://corpus-texmex.irisa.fr/>, 2010.
- [45] “Million song dataset benchmarks,” <http://www.ifs.tuwien.ac.at/mir/msd/download.html>, 2023.
- [46] “Uq video,” <https://github.com/Lsyhprum/WEAVESS/tree/dev/dataset>, 2021.
- [47] “Billion-scale approximate nearest neighbor search challenge: Neurips’21 competition track,” <https://big-ann-benchmarks.com/>, 2021.
- [48] W. Dong, M. Charikar, and K. Li, “Efficient k-nearest neighbor graph construction for generic similarity measures,” in *WWW*, 2011, pp. 577–586.
- [49] C. Fu, C. Wang, and D. Cai, “High dimensional similarity search with satellite system graph: Efficiency, scalability, and unindexed query compatibility,” *TPAMI*, 2021.
- [50] S. Jayaram Subramanya, F. Devvrit, H. V. Simhadri, R. Krishnawamy, and R. Kadekodi, “Diskann: Fast accurate billion-point nearest neighbor search on a single node,” in *NeurIPS*, vol. 32, 2019.
- [51] J. A. V. Muñoz, M. A. Gonçalves, Z. Dias, and R. da Silva Torres, “Hierarchical clustering-based graphs for large scale approximate nearest neighbor search,” *Pattern Recognition*, vol. 96, p. 106970, 2019.
- [52] “Openai embeddings api,” <https://platform.openai.com/docs/guides/embeddings>, 2023.
- [53] “Hugging face embeddings api,” <https://huggingface.co/blog/getting-started-with-embeddings>, 2023.