Data Science and Engineering (2021) 6:280–293
https://doi.org/10.1007/s41019-021-00162-4

Scalable Multi-grained Cross-modal Similarity Query with Interpretability

Mingdong Zhu[1] · Derong Shen[2] · Lixin Xu[1] · Xianfang Wang[1]

Received: 19 January 2021 / Revised: 3 April 2021 / Accepted: 19 April 2021 / Published online: 31 May 2021

Abstract
Cross-modal similarity query has become a highlighted research topic for managing multimodal datasets such as images and texts. Existing researches generally focus on query accuracy by designing complex deep neural network models and hardly consider query efficiency and interpretability simultaneously, which are vital properties of cross-modal semantic query processing system on large-scale datasets. In this work, we investigate multi-grained common semantic embedding representations of images and texts and integrate interpretable query index into the deep neural network by developing a novel Multi-grained Cross-modal Query with Interpretability (MCQI) framework. The main contributions are as follows: (1) By integrating coarse-grained and fine-grained semantic learning models, a multi-grained cross-modal query processing architecture is proposed to ensure the adaptability and generality of query processing. (2) In order to capture the latent semantic relation between images and texts, the framework combines LSTM and attention mode, which enhances query accuracy for the cross-modal query and constructs the foundation for interpretable query processing. (3) Index structure and corresponding nearest neighbor query algorithm are proposed to boost the efficiency of interpretable queries. (4) A distributed query algorithm is proposed to improve the scalability of our framework. Comparing with state-of-the-art methods on widely used cross-modal datasets, the experimental results show the effectiveness of our MCQI approach.

Keywords Cross-modal · Interpretability · Multi-grained · Similarity query ·Scalability

1 Introduction

With rapid development of computer science and technology, multimedia data including images and texts have been emerging on the Internet, which have become the main form of humans knowing the world. Consequently, cross-modal similarity query has been an essential technique with wide applications, such as search engine and multimedia data management. Cross-modal similarity query [1] is such an effective query paradigm that users can get the results of one type by submitting a query of the other type. In this work, we mainly focus on queries between images and texts. For instance, when one user submits a piece of textual description of one football game, most relevant images in datasets can be fetched and vice versa. Cross-modal similarity query should discover latent semantic relationships among different types, it has attracted great interests from researchers.

Due to the significant advantage of deep neural networks (DNN) in feature extraction, DNN models are utilized for cross-modal similarity query [2]. The complex structure and high-dimensional feature maps equip the deep neural networks with considerable power of learning nonlinear relationships; however, at the same time, complex models introduce some drawbacks. First, numerous parameters of deep neural networks make query process and results difficult to be explained. That is, those models have weak interpretability, which is an important property for general and reliable cross-modal query system. Second, in order to find the most similar data objects, typically the cosine similarity

✉ Mingdong Zhu
hackdong@126.com

Derong Shen
shenderong@ise.edu.cn

Lixin Xu
l.xu@hait.edu.cn

Xianfang Wang
wangfang@163.com

1 School of Computer Science & Technology, Henan Institute of Technology, Xinxiang 453703, China

2 School of Computer Science & Engineering, Northeastern University, Shenyang 110819, China

between the high-dimensional feature vector of query object and that of each object in the whole dataset should be computed. Hence, for a large-scale dataset, the computation cost is so high that the query response time will be obnoxious. Existing researches tend to focus on designing complicated composite models to enhance query accuracy and hardly take account of query interpretability, efficiency and scalability at the same time.

Query interpretability of the query framework can improve the credibility of query result. Query efficiency can ensure the accuracy of query result. And query scalability can enhance the adaptability of query methods, especially when faced with large-scale data. Hence, to develop a cross-modal similarity query framework with interpretability, efficiency and scalability is necessary. There are two challenges to achieve this goal. First it is how to bridge the semantic gap among different modality, which need a sophisticated model to capture the common semantic in terms of coarse grain and fine grain. The second challenge is how to enhance interpretability of the query framework with complex structure and millions of parameters. The third challenge is how to integrate the query model with scalability, in case of processing large-scale data, which are ubiquitous nowadays.

Our core insight is that we can leverage deep neural network model to capture multi-grained cross-modal common semantics and build an efficient hybrid index with interpretability and scalability. Hence, in this work, we propose a novel efficient and effective Multi-grained Cross-modal Query framework with Interpretability (MCQI). In order to ensure the adaptability and generality of our framework, during training common feature vectors for different types we first capture coarse-grained and fine-grained semantic information by designing different networks and then combine them. And in order to discover the latent semantic relations between images and texts, we integrated LSTM model and attention model, besides, the data foundation of cross-modal correlative information is constructed in this way. In addition, for the sake of query efficiency, we built an index supporting interpretable query. And further, in order to enhance the scalability of our framework, a distributed query algorithm is proposed based on our framework. At last, to confirm the efficiency and effectiveness of our approach, we systematically evaluate the performances of the approach by comparing with 8 state-of-the-art methods on five widely used multimodal datasets. Concretely, our contributions are shown as follows:

- By integrating coarse-grained and fine-grained semantic learning models, a multi-grained cross-modal query processing architecture is proposed to ensure the adaptability and generality of query processing.
- In order to capture the latent semantic relation between images and texts, the framework combines LSTM and attention mode, which enhances query accuracy for the cross-modal query and constructs the foundation for interpretable query processing.
- Index structure and corresponding nearest neighbor query algorithm are proposed to boost the efficiency of interpretable queries.
- A distributed query algorithm is proposed to improve the scalability of our framework.

The remainder of this paper is organized as follows. Section 2 briefly reviews related work. In Sect. 3, we introduce definitions of problems and then describe in detail our MCQI framework and a *k*NN query algorithm in Sect. 4. Section 5 gives a distributed query algorithm to enhance scalability of our framework. Section 6 provides experimental results and analysis on five datasets, and we conclude in Sect. 7.

2 Related Work

In this section, we briefly review the related researches for cross-modal query, including cross-modal retrieval, latent semantic alignment and cross-modal hashing.

2.1 Cross-modal Retrieval

Traditional methods mainly learn linear projections for different data types. Canonical correlation analysis (CCA) [3] is proposed to learn cross-modal common representation by maximizing the pairwise correlation, which is a classical baseline method for cross-modal measurement. Beyond pairwise correlation, joint representation learning (JRL) [4] is proposed to make use of semi-supervised regularization and semantic information, which can jointly learn common representation projections for up to five data types. S2UPG [5] further improves JRL by constructing a unified hypergraph to learn the common space by utilizing the fine-grained information. Recent years, DNN-based cross-modal retrieval has become an active research topic. Deep canonical correlation analysis (DCCA) is proposed by [6] with two subnetworks, which combines DNN with CCA to maximize the correlation on the top of two subnetworks. UCAL [7] is an unsupervised cross-modal retrieval method based on adversarial learning, which takes a modality classifier as a discriminator to distinguish the modality of learned features. DADN [8] approach is proposed for addressing the problem of zero-shot cross-media retrieval, which learns common embeddings with category semantic information. These methods mainly focus on query accuracy rather than query efficiency and interpretability.

2.2 Latent Semantic Alignment

Latent semantic alignment is the foundation for interpretable query. [9] embeds patches of images and dependency tree relations of sentences in a common embedding space and explicitly reasons about their latent, intermodal correspondences. Adding generation step, [10] proposes a model which learns to score sentence and image similarity as a function of R-CNN object detections with outputs of a bidirectional RNN. By incorporating attention into neural networks for vision related tasks, [11, 12] investigate models that can attend to salient part of an image while generating its caption. These methods inspire ideas of achieving interpretable cross-modal query, but neglect issues of query granularity and efficiency.

2.3 Cross-modal Hashing

Deep cross-modal hashing (DCMH) [13] combines hashing learning and deep feature learning by preserving the semantic similarity between modalities. Correlation auto-encoder hashing (CAH) [14] embeds the maximum cross-modal similarity into hash codes using nonlinear deep autoencoders. Correlation hashing network (CHN) [15] jointly learns image and text representations tailored to hash coding and formally controls the quantization error. Pairwise relationship guided deep hashing (PRDH) [16] jointly uses two types of pairwise constraints from intra-modality and intermodality to preserve the semantic similarity of the learned hash codes. [17] proposes a generative adversarial network to model cross-modal hashing in an unsupervised fashion and a correlation graph-based learning approach to capture the underlying manifold structure across different modalities. For large high-dimensional data, hashing is a common tool, which can achieve sublinear time complexity for data retrieval. However, after constructing a hash index on hamming space, it is difficult to obtain flexible query granularity and reasonable interpretability.

2.4 Distributed Similarity Query

Existing methods for distributed similarity queries in metric spaces can be partitioned into two categories [18]. The first category utilizes basic metric partitioning principles to distribute the data over the underlying network. [19] proposes a distributed index, GHT* index, which can exploit parallelism in a dynamic network of computers by putting a part of the index structure in every network node. [20] proposes a mapping mechanism that enables to actually store the data in well-established structures such as the B-tree. The second category utilizes the index integrating technique to distribute the data. Paper [21] integrates R-tree and CAN overlay to process multi-dimensional data in a cloud system. Paper [22] combines B-tree and BATON overlay to provide a distributed index which has high scalability but incurs low maintenance. They both choose a part of local index nodes to build global index node by computing the cost model. [23] integrates quadtree index with Chord overlay to enable more powerful accesses to data in P2P networks. In this paper, we adopt the pivot-mapping-based method due to two reasons below. These methods are not enough due to two reasons below. First, they are query sensitive; that is, they cannot adjust distribution of data for different query load and then cannot keep load balance, which is also important for distributed environment.

3 Problem Description

For cross-modal similarity query, given a query object of one type, most similar objects of the other type in the dataset should be returned. The formal definition is shown below.

The multimodal dataset consists of two modalities with m images and n texts, which is denoted as $D = \{D^t, D^i\}$. The texts are encoded as a one hot code originally and in the set D the data of text modality are denoted as $D^t = \left\{x_k^t\right\}_{k=1}^m$, where the kth text object is defined as $x_k^t \in R^{l_k * c}$ with the sentence length l_k and the vocabulary size c. The data of image modality are denoted as $D^i = \left\{x_k^i\right\}_{k=1}^n$, where the kth image instance is defined as $x_k^i \in R^{w*h*c'}$ with image resolution $w*h$ and color channel number c'. Besides, the pairwise correspondence is denoted as $(\mathrm{x}_k^t, \mathrm{x}_k^i)$, which means that the two instances of different types are strongly semantically relevant. Cross-modal similarity query means that given one query object it is to find similar objects of the other modality which share relevant semantics with the given one, kNN query is a classical type of similarity query and the definition is given as follows.

Definition 1 (kNN Query). Given an object q, an integer $k > 1$, dataset D and similarity function SIM, the k nearest neighbors query kNN computes a size-k subset $S \subseteq D$, s.t. $\forall o_i \in S, o_j \in D - S : SIM\left(q, o_i\right) \geq SIM\left(q, o_j\right)$. In this work, we set cosine similarity as similarity function.

Table 1 lists the used notations throughout this paper. The list mainly consists of the notations which are mentioned far from their definitions.

4 Proposed Model

In this section, we describe the proposed MCQI framework in detail. As shown in Fig. 1, MCQI framework consists of two stages. The first stage is the learning stage, which models common embedding representation of multimodal data by fusing coarse-grained and fine-grained semantic information. The

Table 1 Used notations

Notation	Description
SIM	similarity function
(x_k^t, x_k^i)	the *k*th matched pair of images and texts
d_l	dimension of local common embedding space
d_g	dimension of global common embedding space
TU	the set of patch relation tuples between images and texts
Ins_i	the *i*th data instance in the dataset
$CFVF_i$	the *i*th common fine-grained semantic feature
$CFVC_i$	the *i*th common coarse-grained semantic feature
δ	weight factor to balance fine-grained and coarse-grained features
C	number of computing nodes
σ	probability of weight factor can be omitted

second stage is the index construction stage, in which M-tree index and inverted index are integrated to process efficient and interpretable queries. In the following paragraphs, we introduce it in the aspects of embedding representations of multimodal data and interpretable query processing.

4.1 Embedding Representations of Multimodal Data

In the first stage, MCQI learns the embedding representation of multimodal data by fusing coarse-grained and fine-grained semantic information.

4.2 Fine-grained Embedding Learning with Local Semantics

Different methods are used to extract local semantic features for texts and images. For texts, EmbedRank [24] is utilized to extract keyphrases. Then, a pretrained model Sent2vec[25] is chosen for computing the embedding of each keyphrase. Then, by three-layer fully connected neural network, we map each keyphrase into the common embedding space with dimension *d_l*, denoted as ts_{pq}, which means the embedding representation of the *q*th keyphrase of the *p*th text description.

For images, Region Convolutional Neural Network (RCNN) [26] is utilized to detect objects in images. We use top detected locations of the entire image as local semantic features and then compute the common embedding vectors based on the visual matrix in each bounding box by a pretrained convolutional neural network and by transition matrix transform the vector to common space with dimension *d_l*; lastly, we get is_{uv}, which means the embedding representation of the *v*th bounding box of the *u*th image.

Typically, for a pair of matched text and image, at least one of keyphrases in the text is semantically relevant with a certain bounding box in the image instance; that is, at least one common embedding vector of the text instance is close to a certain common embedding vector of the image instance. Base on this intuitiveness, according to hinge rank loss function, we set the original objective of fine-grained embedding learning as follows:

$$C_b(ts_{pq}, is_{uv}) = \sum_{q}\sum_{v}\left(\left(\frac{Pnum}{Anum}\right)^{I(p\neq u)}\left(1-\frac{Pnum}{Anum}\right)^{I(p=u)}\max\left(0, M - (-1)^{I(p\neq u)}\frac{ts_{pq}\cdot is_{uv}}{|ts_{pq}|\cdot|is_{uv}|}\right)\right) \quad (1)$$

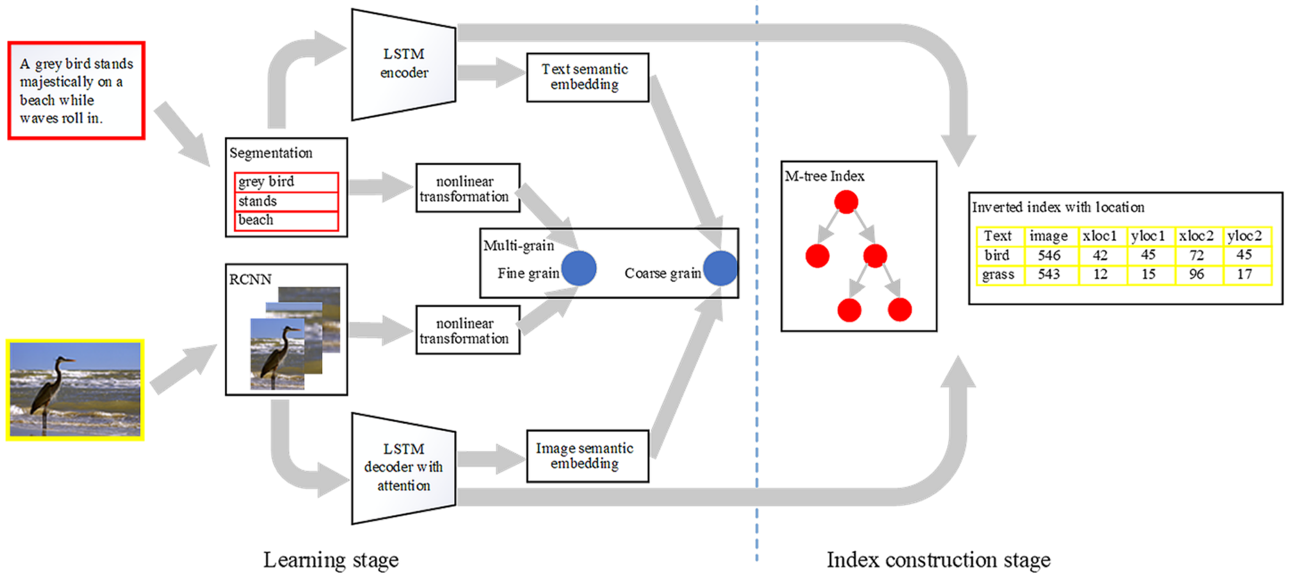

Fig. 1 The framework of MCQI

Here, *Pnum* is the number of matched pairs in the training sample set and *Anum* is the training sample capability. $\left(\alpha\frac{Pnum}{Anum}\right)^{I(p=u)}\left(1-\alpha\frac{Pnum}{Anum}\right)^{I(p\neq u)}$ is utilized to balance positive and negative samples. $\frac{ts_{pq}\cdot is_{uv}}{|ts_{pq}|\cdot|is_{uv}|}$ is the cosine similarity of two embedding vectors. M is the margin constant, which defines the tolerance of true positive and true negative. The more M is close to 1, the stricter it is about semantically recognizing true positive and true negative.

C_b cost is computed over all pairs of local features between text instances and image instances. However, in many cases, for two semantically relevant instances only few parts of local features are matched and similar pairs are difficult to be acquired by computation over all pairs. To address this problem, according to MILE [9], we make a multiple instance learning extension of formula (1) as shown in formula (2). For each text instance, local features of matched image are put into a positive bag, while local features in other image are treated as negative samples.

Our implementation of LSTM with soft attention is based on [11]. $\boldsymbol{a_i}$ is the input, $\boldsymbol{y}$ and α_{ti} are outputs, $\boldsymbol{y}$ is the generated sequential text and α_{ti} represents importance of feature vector a_i when generating the tth word. please note that each word y_t has an attention weight α_{ti} for each feature vector a_i and each tuple $tu_t = <y_t$, imageID, α_{ti}, $xloc_i$, $xloc1_i$, $xloc2_i$, $yloc1_i$, $yloc2_i>$ is stored for answering future queries, where imageID is the image's unique identifier, $xloc_i$, $xloc1_i$, $xloc2_i$, $yloc1_i$, $yloc2_i$are the corresponding coordinate position of a_i in the image. We collect all tuples as set $TU=\{tu_t\}$.

For generated sequential text y, Universal Sentence Encoder is utilized to generate the coarse-grained representative vector of y, denoted as *GIV*, while the coarse-grained representative vector of original paired training text by Universal Sentence Encoder is denoted as *OTV*.

$$C_P = \min_{kqv}\sum\nolimits_q\sum\nolimits_v\left(\frac{Pnum}{Anum}\right)^{I(p\neq u)}\left(1-\frac{Pnum}{Anum}\right)^{I(p=u)}\max\left(0, M-k_{qv}\frac{ts_{pq}\cdot is_{uv}}{\left|ts_{pq}\right|\cdot\left|is_{uv}\right|}\right)$$
$$\text{s.t.}\sum_{\mathrm{v}\in \mathrm{B_q}}\left(\mathrm{k_{qv}}+1\right)\geq 2\forall\mathrm{v}, k_{qv}=\begin{cases}1, p=u\\-1, p\neq u\end{cases} \tag{2}$$

Here, $\mathrm{B_q}$ is the positive bag of the qth feature vector, $\mathrm{k_{qv}}$ is the correlation index which indicates whether the corresponding text instance and image instance are matched. It is worth notice that each feature vector $\mathrm{is_{uv}}$ and the corresponding bounding box are stored in the storage system for processing interpretable queries.

4.3 Coarse-grained Embedding Learning with Global Semantics

Coarse-grained embedding network tries to capture global common semantics between texts and images. For texts, Universal Sentence Encoder [27] is utilized to extract feature vectors of texts and by fully connected layers the feature vectors are transformed into the global common embedding space with dimension *d_g*.

For images, inspired by [11] that pretrained LSTM with soft attention model is integrated to translate images into sequential representation. For an image, feature maps before classification in a pretrained R-CNN network and the whole image's feature maps before fully connected layers in pretrained CNN network are combined into feature vectors, denoted as $a=\left\{a_i\right\}_{i=1}^{LV}$ and LV is the number of the feature vectors.

Intuitively, global training objective function is shown as follows.

$$C_G = GIVOTV \tag{3}$$

4.4 Multi-grained Objective Function

We are now ready to formulate the multi-grained objective function. The objective function is designed by two criteria. First, it is likely that matched pairs of images and texts have similar patches, which applies to C_P. Second, matched pairs of image and text probably have similar global semantics, which applies to C_G. By integrating C_P and C_G, the objective function is defined as follows.

$$C(\theta)=\alpha C_P(\theta)+\beta C_G(\theta)+\gamma|\theta|_2^2, \tag{4}$$

where θ is a shorthand for parameters of our model and α, β, γ are hyperparameters which are computed by cross-validation. $|\theta|_2^2$ is the regularization.

4.5 Optimization

The proposed model consists of two branches, which are designed for common fine-grained semantic and coarse-grained semantic, respectively. Naturally, the training process is divided into two stages, i.e., branch training and joint training. Both training processes are based on stochastic gradient descent (SGD) with a batch size of 32, a momentum of 0.9 and a weight decay of 0.00005.

Stage 1: In this stage, branches for common fine-grained semantic and coarse-grained semantic are trained in turn, taking formula (2) and formula (3) as loss functions, respectively. In the fine-grained branch, pretrained Sent2Vec model and RCNN model are utilized, while in the coarse-grained branch, pretrained several pretrained Universal Sentence Encoder model and LSTM model are utilized. The default parameters in those pretrained models are utilized, and its parameters are kept fixed at this stage. The other parameters of our model, including the attentional mechanism, are automatically initialized with the Xavier algorithm [28].

Stage 2: After all branch networks are trained, we jointly fine-tune the entire model parameters by combining the loss terms over all granularities in formula (4).

4.6 Interpretable Query Processing

In MCQI framework, images and texts can be represented by high-dimensional feature vectors, which include fine-grained and coarse-grained semantic features. Denote IFV_i as feature vectors of the ith instance Ins_i, then $IFV_i = \{CFVF_i, CFVC_i\}$, where $CFVF_i$ and $CFVC_i$ mean the corresponding common fine-grained semantic feature and the coarse-grained semantic feature of Ins_i, respectively. Given a query instance, i.e., an image or text instance, in order to find the matched cross-modal instance, i.e., the most relevant text or image instance, the similarity between two cross-modal instances can be computed by cosine similarity shown in formula (5) as follows.

$$\mathrm{SIM}\left(Ins_i, Ins_j\right) = \delta \frac{CFVF_i \cdot CFVF_j,}{\left|CFVF_i\right| * \left|CFVF_j\right|} + (1-\delta)\frac{CFVC_i \cdot CFVC_j,}{\left|CFVC_i\right| * \left|CFVC_j\right|} = \delta \mathrm{Cos}ine\left(CFVF_i, CFVF_j\right) + (1-\delta)\mathrm{Cos}ine\left(CFVC_i, CFVC_j\right) \tag{5}$$

Here, Ins_i and Ins_j are two cross-modal instances, δ is the weight factor, Cosine is the cosine similarity function.

A naive method to obtain the matched cross-modal instances is pairwise computation; however, this method is inefficient. Particularly when the dataset is large and the dimension of vectors is high, the computation is nontrivial. To address this, an inverted index and an M-tree index are integrated into MCQI model. The M-tree index increases the efficiency of queries and the inverted index enhances the interpretability of queries. Index construction and query processing method based on the indices are discussed separately as follows.

4.7 Index Construction

It is shown in formula (5) the similarity between two instances mainly is calculated by the cosine similarity of two types of feature vectors. By assuming that variables obey uniform distribution, we get Observation 1 in the following. Observation 1 shows that cosine similarity between the whole feature vectors of Ins_i and Ins_j is close to SIM(Ins_i, Ins_j).

Observation 1 For Random Variable $\delta \in [0.2, 0.8], \exists \varepsilon, \sigma \in [0, 1]$, s.t. $P\left(\left|\left(\delta\frac{CFVF_i \cdot CFVF_j,}{\left|CFVF_i\right| * \left|CFVF_j\right|} + (1-\delta)\frac{CFVC_i \cdot CFVC_j,}{\left|CFVC_i\right| * \left|CFVC_j\right|}\right) - \frac{IFV_i \cdot IFV_j,}{\left|IFV_i\right| * \left|IFV_j\right|}\right| < \varepsilon\right) > \sigma$, i.e., $P\left(\left|SIM\left(Ins_i, Ins_j\right) - \mathrm{Cos}ine\left(Ins_i, Ins_j\right)\right| < \varepsilon\right) > \sigma$.

This Observation is obtained by statistical hypotheses testing method, which will be illustrated in the experiments. By setting $DIF = \left|\mathrm{SIM}(\mathrm{Ins_i}, \mathrm{Ins_j}) - \mathrm{Cosine}(\mathrm{Ins_i}, \mathrm{Ins_j})\right|$, we get $\mathrm{P}(DIF < \varepsilon)) > \sigma$. In experiments, when set $\varepsilon = 0.05$, we have $\sigma = 0.9$ and when set $\varepsilon = 0.1$, we have $\sigma = 0.98$.

It is known that the M-tree is an efficient structure for NN queries in metric spaces. In order to use M-tree index, cosine distance should be transformed to angular similarity (AS) which is metric. The angular similarity between Ins_i and Ins_j is defined in formula (6) in the following.

$$\mathrm{AS}\left(Ins_i, Ins_j\right) = 2\frac{\arccos(\mathrm{Cosine}(\mathrm{Ins_i}, \mathrm{Ins_j}))}{\pi} \tag{6}$$

Lemma 1. *For any instance q, the nearest neighbor of q by angular similarity is also the nearest neighbor of q by cosine similarity.*

Lemma 1 can be easily proved by contradiction, which is omitted for simplicity.

Based on Lemma 1 and formula (6), an M-tree is constructed on the data set of feature vectors. And then M-tree is augmented with an inverted index of semantic relationship tuple set TU, which is mentioned in Sect. 4.1.

4.8 Interpretable kNN Query

For processing similarity queries efficiently, we adopt a filter-and-fine model. Our method first obtains candidates of matched objects by M-tree and then verifies the candidates and identifies the final answers.

The M-tree inherently supports range query, denoted as Range (Ins_i, r), where Ins_i is the query instance and r is the query range. In our algorithm the kNN candidates can be efficiently obtained by two range queries on M-tree. To verify the candidates, formula (5) is utilized and for the verified objects, Inverted index is accessed to give reasons why the objects are relevant to the query. The detailed query processing is shown in algorithm 1 as follows. Specifically, at first we use range query Range (Ins_i, 0) to find the closest index node and read all the objects in the node(line 2). If the number of objects is less than k, we read its sibling nodes through its parent node, recursively, until we obtain k objects (line 3). And then we use the kth farthest distance r from the query instance to issue the second range query by setting range as r and get the candidates. Finally, we utilized formula (5) to verify the candidates and each matched pair is augmented with the relationship interpretation through inverted index (line 6–8).

```
Algorithm 1 : kNN Query Processing
Input: NN($Ins_i$, $k$)
Output: Result set $R$
1   $R = \emptyset$
2   $cn$= Range($Ins_i$, 0)
3   Get at least $k$ objects from in or siblings or parents recursively
4   Set $r$ to the farthest distance
5   $S$= Range($Ins_i$, 2*$r$)
6   Verify $S$ by formula (5)
7   Set $R$ to top $k$ similar instances of verified objects
8   Augmented $R$ with interpretation by inverted index
9   Return $R$
```

As for complexity, considering the first range query with range zero, the cost of processing a query is O(H), where H is the height of the M-tree. As for the second range query the selectivity of a range query is se, the cost of each level of index nodes can be approximated as a geometric sequence with common ratio, $cr*se$, where cr is the capacity of index node. Hence, the average cost is:

$$\frac{\mathrm{cr*se*(1 - (cr*se)^H)}}{\mathrm{1 - cr*se}} \tag{7}$$

As for query accuracy, by Observation 1 and Lemma 1, we can get Observation 2 as follows.

Observation 2. Algorithm 1 can obtain kNN instances of the query instances with probability more than σ.

Proof. We assume o^* is the actual the kth NN query result but is not returned. Denote dis is the distance between the returned kth NN query result and the query. and by Lemma 1, we can get that the distance between o^* and the query is less than $dis + DIF$. And set $\varepsilon = dis$, according to Observation 1, by probability σ or more, DIF is less than dis. So, by algorithm 1, we can get the query result o^*, which is a contradiction for assumption. Then the Observation 2 is proved.

5 Distributed Algorithm

When the data set is relatively large, the computational complexity of the algorithm will be relatively high as shown in formula 7. Therefore, in order to effectively process large-scale data sets, this section will extend the framework to a distributed environment and propose a distributed kNN algorithm.

The distributed algorithm is based on the idea of divide and conquer. Each computing node in a P2P distributed system is independent and autonomous. Let C be the number of computing nodes, PV is the pivot set of the data set, $PV = \{pv_i\}$, where $1 \le i \le pn$ and pn is the number of pivot points. PV is stored on each computing node as global information. Each computing node is responsible for one or more pivot points. Data are divided into computing nodes according to the distance between the data object and the pivot point. Then, each computing node builds M-tree and inverted index locally. When a computing node receives a similarity query q with query range R, the computing node will act as the coordinator and calculate the relevant pivot point by formula 8.

$$\mathrm{SIM}(pv_i, \overline{q}) \ge 1 - (R + maxd_i), \tag{8}$$

where $maxd_i$ is the largest distance among the data objects maintained by pv_i. Then, the coordinator will forward the query to the computing node where the relevant pivot point is located and each computing node will calculate the query result through the local indices and return it to the coordinator. Finally, the coordinator collects the intermediate query results and returns the final result to the user. Obviously, the selection of pivot points and query algorithms are the key points of query performance and these two parts will be discussed in detail as follows.

5.1 Selection of Pivot Points

The main function of the pivot point is to filter irrelevant data objects in the query process, so the index of selecting the pivot point is to increase the filtering ability of the query as much as possible. In the metric space, the farthest pivot point is generally selected. Based on this heuristic method, we propose a pivot point selection scheme similar to [29]. First randomly select a data object *o* from the sample data set and then put the data object farthest from *o* in the sample data set to the pivot point set *PV*, then further add the data object with the largest average distance between the sample data set and the central pivot point to the *PV* and then repeat the previous step until |*PV*|=*pn*.

5.1.1 Query-Sensitive Load Balancing

In a distributed environment, consistent hashing is used to maintain and manage the pivot point, that is, the pivot point is divided into [0, 2^{max}] domain, [0, 2^{max}] is divided into multiple intervals (token) and each compute node is responsible for one or more intervals and the query is routed through the distributed hash table in the system. Through a hash method such as SHA1, the pivot point will be divided evenly on each computing node. However, the query load is not always evenly distributed and the distribution will change dynamically. Therefore, in order to achieve the load balance of the system, a query-aware adjustment method is needed. First, set the threshold *t* for load balance. If a computing node (ComputerA) exceeds *t* times the average load, that is, ComputerA becomes a query bottleneck of load, then ComputerA communicates with the adjacent computing node (ComputerB) of its responsible area, then reduce the area that ComputerA is responsible for while increase the area that ComputerB is responsible for and move the corresponding pivot point from ComputerA to ComputerB. After the last step, if other computing nodes have a load balance problem, repeat this process for this computing node until the load balance of the system is achieved. Note that in order to avoid thrash, the load adjustment method should be performed in the same direction.

5.1.2 Computation of pn

The execution time of query processing can be divided into two parts, one part is the time *gt* for computing the relevant computing node based on the pivot point and the other part is the time *lt* for each computing node to perform a local query. Therefore, the computing time for query processing is $ct = gt + lt$. Obviously, the computing time *gt* of the relevant computing node is proportional to the number of pivot points *pn*, that is, $gt = \alpha * pn$, α is the coefficient ratio and α is related to the processing capability of computing nodes. The computing time of the query *lt* and *pn* is inversely proportional, in the average, $lt = \beta \frac{r^{\left(\log_m \frac{N}{pn}\right)}}{r-1} - 1 = \beta \frac{\left(\frac{N}{pn}\right)^{\left(\frac{1}{\log_r m}\right)}}{r-1} - 1$, where *r* is the average selection degree of the child nodes of the index tree by the query, *m* is the out degree of the index tree, *N* is the size of the data set, β is the coefficient ratio and β is determined by the average processing capability of the computing node. Therefore, the formula of computing *ct* can be obtained:

$$\mathrm{ct} = \alpha * pn + \beta \frac{\left(\frac{\mathrm{N}}{\mathrm{pn}}\right)^{\left(\frac{1}{\log_r \mathrm{m}}\right)} - 1}{\mathrm{r - 1}} \tag{9}$$

By deriving and calculating the extreme value, it is easy to get when

$$pn = \left(\frac{\beta N^{\frac{1}{\log_r m}}}{\alpha (r-1) \log_r m} \right)^{\frac{\log_r m}{\log_r m + 1}}, \tag{10}$$

ct takes the minimum. By Formula 10, the number of pivot points can be obtained.

5.2 Distributed kNN Query Algorithm

As mentioned at the beginning of this section, by Formula 8 it is easy to handle range queries. In this section, we discuss distributed nearest neighbor query algorithm.

Considering a simple case, when $k=1$, it is a 1NN query. When a computing node receives a 1NN query *q*, the computing node as the scheduler first initiates a query object *q* and the query radius 0. Calculate the relevant pivot points, that is, calculate the pivot point set $PS = \{pn_i | \mathrm{SIM}(pn_i, q) \geq 1\text{-}maxd_i\}$, where $maxd_i$ is the distance between the pivot node pn_i and the farthest data object maintained. Then, the scheduler forwards the query to the computing node (denoted as CS) responsible for the data objects in the PS set and each computing node calculates the local NN of the data object *q* and returns it to the scheduler. After receiving all candidate nearest neighbors, the scheduler calculates the data object with the smallest distance to *q* and let *mind* be the smallest distance. After that, the scheduler uses *q* as the query object and *mind* as the query range to calculate the relevant pivot points and forwards the NN query to the computing nodes responsible for these pivot points except for the CS set. Finally, the scheduler collects candidate data objects to calculate the NN and returns it to the user.

The *k*NN query algorithm is discussed as follows. The specific process is shown in Algorithm 2. First, the initial query distance *initR* is estimated according to the statistical histogram and the relevant computing node (line 2) is calculated. For a kNN query with a query object of q, The distance between *q* and each pivot point is $qdist_i = 1\text{-SIM}(pn_i, q)$, let

$$initR = \text{argmin}_r \cdot \sum_{i=1}^{i \le pn} \text{NumHist}(\text{pv}_i, \text{qdist}_i - r, \text{qdist}_i + r) \le k \tag{11}$$

where NumHist(*r*, *dmin*, *dmax*) function obtain the number of data object in index with root *r* between *dmin* and *dmax*. Then, forward the query to the relevant computing node set CS_1. Each computing node calculates the local *k*NN data object as a candidate set and returns it to the scheduler. The scheduler calculates the smallest *k*NN candidate data object distance *mind* from all the candidate data objects and calculate the relevant computing node CS_2 (lines 3–7) when the query range is *mind*, forward the request to the computing node set CS_1- CS_2 and collect the local *k*NN candidate data objects of each computing node, calculate the final *k*NN result and return (lines 8–11).

The *k*NN query algorithm is also implemented using two range queries, but the main difference is that in the first range query, *k*NN uses the histogram information summarized by the pivot point to predict a better query range *initR*. *initR* can have a good estimate of the *k* nearest neighbor data objects, thereby effectively reducing the cost of the second range query.

Algorithm 2: Distributed *k*NN Query Processing

```
Input: DNN(Ins_i, k)
Output: Result set R
1   R = ∅
2   Compute initR by formula (11), and then calculate related computing nodes CS_1
3   Forward the query q and query radius initR to related computing nodes
4   candit_y =each computing node y∈CS_1 computes the local k nearest neighbors
5   candit_1 =collect candidate kNN result from all relevant compute nodes
6   mind= calculate the distance between the k-th nearest neighbor in candit_1 and q
7   CS_2 = calculate the relevant computing nodes when the query radius is mind
8   Forward the query q to the set of computing nodes CS_2 - CS_1
9   candit_y =each computing node y∈CS_2 computes the local k nearest neighbors
10  candit_2 =collect candidate kNN result from all relevant compute nodes
11  Return kNN result from candit_1 and candit_2
```

6 Experiment

6.1 Experiment Setup

We evaluate our cross-modal query performance on Flickr8K [30], Flickr30K [31], NUS-WIDE [32], MS-COCO [33] and a synthetic dataset Synthetic9K in our experiments. Flickr8K consists of 8096 images from the Flickr.com website and each image is annotated by 5 sentences by Amazon Mechanical Turk. Flickr30K is also a cross-modal dataset with 31,784 images including corresponding descriptive sentences. NUS-WIDE dataset is a web image dataset for media search, which consists of about 270,000 images with their tags and each image along with its corresponding tags is viewed together as an image/text pair. MS-COCO contains 123,287 images, and each image is also annotated by five independent sentences provided by Amazon Mechanical Turk. By extracting 2000 image/text pairs from each above dataset, we obtain a hybrid dataset, denoted as Synthetic9K. For each data set, 10% data are used as testing set and validation set, while the rest are training set.

We compare our approach with eight state-of-the-art cross-modal retrieval methods, including CCL [34], HSE [35], DADN [8], SCAN [36], DCMH [13], LGCFL [37], JRL [4], KCCA[38]. CCL learns cross-modal correlation by hierarchical network in two stages. First, separate representation is learned by jointly optimizing intra-modality and intermodality correlation and then a multi-task learning is adopted to fully exploit the intrinsic relevance between them. HSE proposes a uniform deep model to learn the common representations for four types of media simultaneously by considering classification constraint, center constrain and ranking constraint. DADN proposes a dual adversarial distribution network which takes zero-shot learning and correlation learning in a unified framework to generate common embeddings for cross-modal retrieval. SCAN considers the latent alignments between image regions and text words to learn the image-text similarity. DCMH combines hashing learning and deep feature learning by preserving the semantic similarity between modalities. LGCFL uses a local group-based priori to exploit popular block based features and jointly learns basis matrices for different modalities. JRL applies semi-supervised regularization and sparse regularization to learn the common representations. KCCA follows the idea of projecting the data into a higher-dimensional feature space and then performing CCA. Some compared methods rely on category information for common representation learning, such as CCL and HSE; however, the datasets have no label annotations available. So, in our experiments first keywords are extracted from text descriptions by TF-IDF method and seen as labels for corresponding images. For distributed query processing, our algorithms are compared with two most related methods. One is a naive method(SimD), data objects are scattered randomly and when there is a query, objects are compared with query in pairwise way. The other is a state-of-the-art method [39](DistMP), which is a general framework based on MapReduce.

Following [34], we apply the mean average precision (MAP) score to evaluate the cross-modal query performance. We first calculate average precision (AP) score for each query in formula (8) and then calculate their mean value as MAP score.

$$AP = \frac{1}{|R|} \sum_{i=1}^{k} p_i * rel_i \tag{12}$$

where |*R*| is the number of ground-truth relevant instances, k is from the *k*NN query, p_i denotes the precision of the top *i* results and rel_i is the indicator whether the *i*th result is relevant.

We adopt TensorFlow [40] to implement our MCQI approach. In the first stage, we take 4096 dimensional feature extracted from the image inside a given bounding box from RCNN. For the nonlinear transformation model, we use three fully connected layers with 1,024 dimensions and set the dimension of common embedding space *d_l* and *d_g* as 1024. The Sent2vec for fine-grained semantics has 700 dimensions, which is pretrained on Wikipedia and Universal Sentence Encoder for coarse-grained semantics has 512 dimensions. Experiments for centralized algorithms are conducted on a server with Intel E5-2650v3, 256 GB RAM, NVIDIA V100 and Ubuntu 16.04 OS, while experiments for distributed algorithms are processed on a cluster of 30 computer nodes with Intel Core i5-10210 1.6 GHz*4CPU and 8 GB memory.

6.2 Verification of Observation 1

Figures 2 and 3 show the accuracy of $DIF < 0.05$ and $DIF < 0.1$ respectively, with different sample size. δ is randomly generated from three different ranges, i.e., [0.2, 0.8], [0.3, 0.7], [0.4, 0.6] and for different varying ranges, it can tell that when δ is closer to 0.5 the accuracy is higher. In the situation of $DIF < 0.05$, with the increasing of sample size, the accuracy is steadily more than 0.9. And for $DIF < 0.1$, with the increasing of sample size, the accuracy is steadily more than 0.99. Without loss of generality, according to statistical hypotheses testing method, in the situation of δ=[0.2, 0.8], we assume $DIF < 0.05$ with significant level as 0.1. In our experiments with sample size 100,000, the mean value of *DIF* is 0.021, sample variance is 0.00045 and because the standard deviation is unknown, t-distribution should be referred. Test statistic is -0.63. And with significant level 0.1, the critical quantile is -1.28. because -0.63 > -1.28, the assumption is accepted.

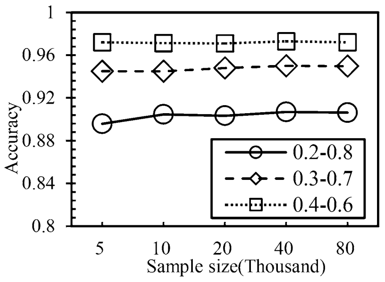

Fig. 2 Accuracy of $DIF < 0.05$

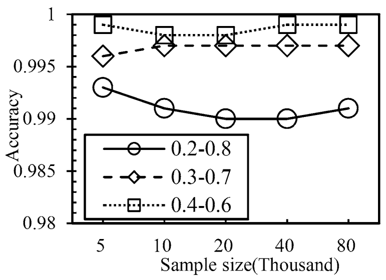

Fig. 3 Accuracy of $DIF < 0.1$

6.3 Performance of Query Accuracy

We present query accuracy of our MCQI approach as well as all the compared methods in this part. Table 2 shows the MAP scores for 30NN query. As shown in the table, the accuracies of DNN-based methods like DADN and CCL are higher than traditional methods on average. Due to the fusion of multi-grained semantic feature and transfer learning embedding, MCQI approach steadily achieves the best query accuracies. The number of data categories in

Table 2 MAP scores of MCQI and compared methods for 30NN query

	Flickr8K	Flickr30K	NUS-WIDE	MS-COCO	Synthetic9K
CCL	0.518	0.566	0.649	0.623	0.423
HSE	0.527	0.526	0.596	0.677	0.452
DADN	0.615	0.412	0.438	0.692	0.341
SCAN	0.223	0.191	0.218	0.254	0.136
DCMH	0.509	0.432	0.487	0.488	0.392
LGCFL	0.457	0.424	0.495	0.369	0.346
JRL	0.421	0.543	0.583	0.576	0.339
KCCA	0.367	0.394	0.421	0.345	0.306
MCQI	**0.534**	**0.683**	**0.712**	**0.702**	**0.528**

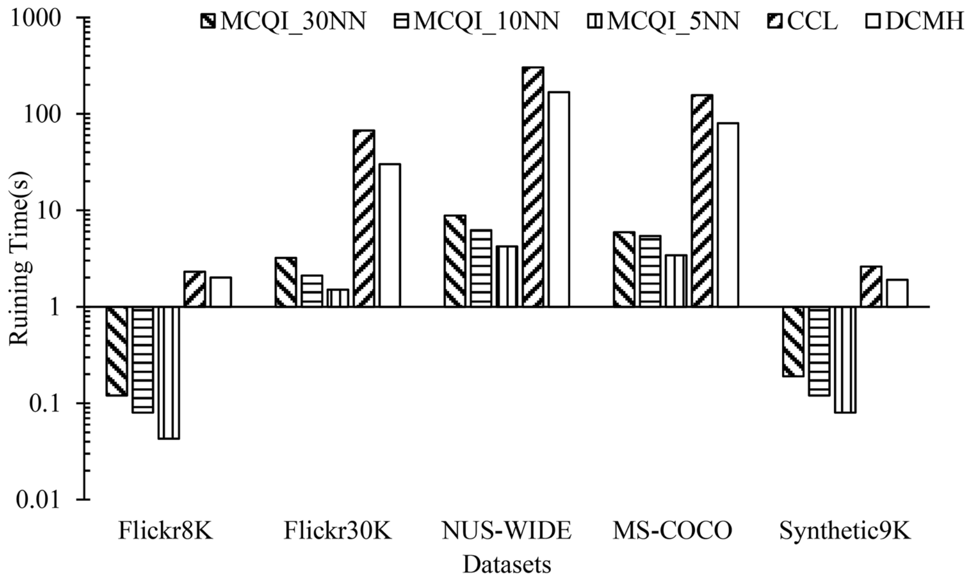

Fig. 4 Performance of query time

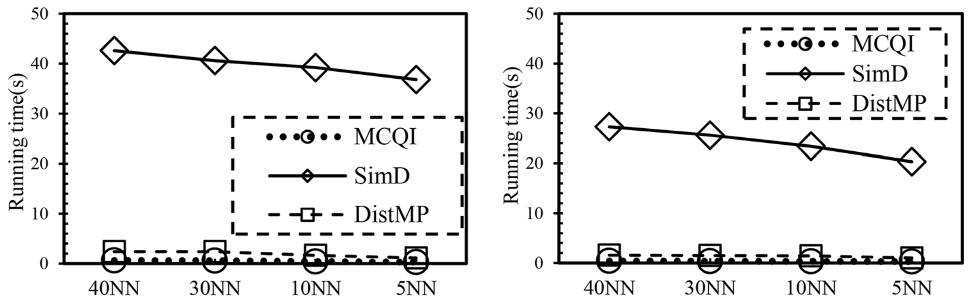

Fig. 5 Effect of distributed query on NUS-WIDE. Effect of distributed query on MS-COCO

Sythetic9K is more than other datasets, and comparatively learning common semantic embeddings are more dependent on the quantity of training data. So, under the same condition, the accuracy is impacted relatively.

6.4 Performance of Query Time

As shown in Fig. 4, we measure the query time for our proposed MCQI approach as well as two representative methods on 5 datasets. CCL is a DNN-based method and DCMH is a hash-based method. For CCL pairwise computation is need to get *k*NN result. And for DCMH, data can be transformed into binary code and it is fast to obtain 1NN, while for *k*NN with varying *k*, query time is affected. Intuitively query times are proportional to the size of the datasets. As CCL and DCMH are not very sensitive to *k* of *k*NN queries, we show query time of only 30NN queries on each dataset. From 30NN queries to 5NN queries, filtering effect of M-tree index enhances, consequently query times decrease. In all cases, MCQI is fastest among the methods. Particularly for 5NN, average running times for MCQI are about 13 times faster than that of CCL and 20 times faster than DCMH, i.e., our approach on average outperforms CCL and DCMH by an order of magnitude.

6.5 Performance of Distributed Algorithm

In order to show the scalability of our framework. Figure 5 present the running time of methods with varying *k* of *k*NN query on NUS-WIDE and MS-COCO dataset, respectively. In terms of running time, MCQI is nearly three times as fast as DistMP, which is one order of magnitude faster than SimD. SimD as a pairwise method causes enormous communication cost in distributed environment, while DistMP which utilizes the metric distance to filter unrelated data can save computation cost. However, for DistMP the lack of an efficient index leads to worse query performance than MCQI. In essence, MCQI is composed of two rounds of NN query groups and it is easy to see that MCQI is significantly better than SimD and DistMP.

6.6 Query Interpretability

Figure 5 shows some examples of cross-modal similarity query results. Because MCQI not only contains the latent

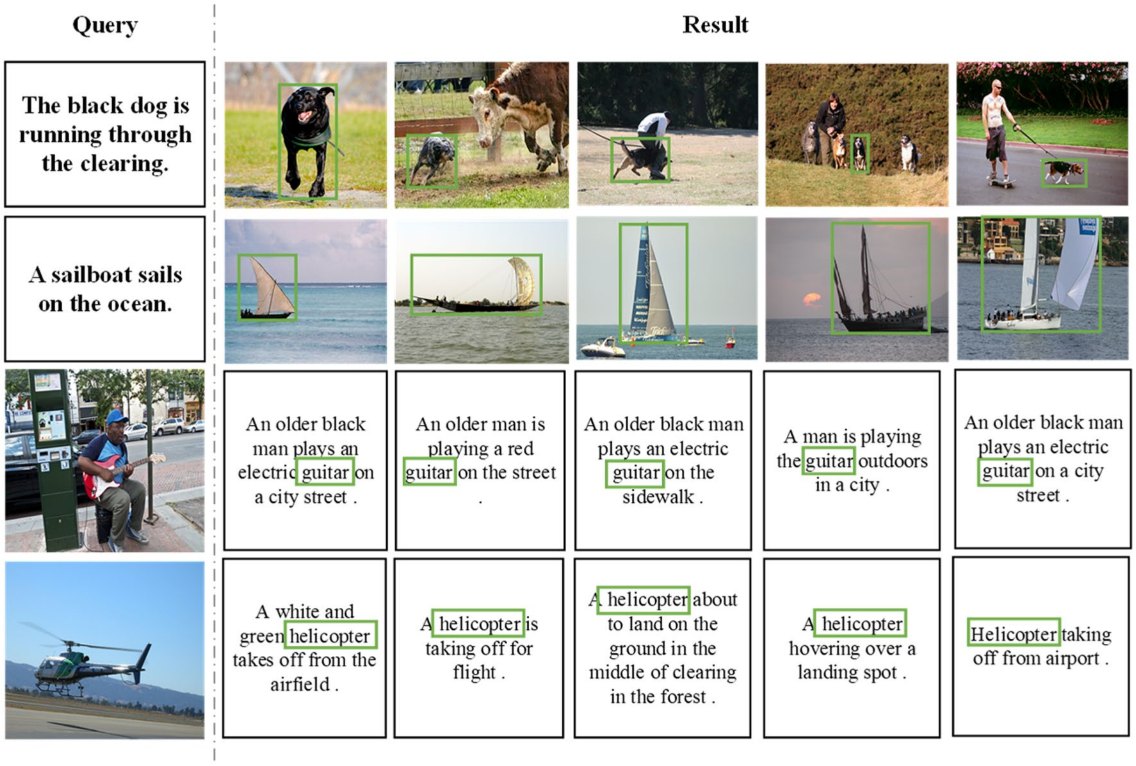
Query
Result
The black dog is running through the clearing.
A sailboat sails on the ocean.
An older black man plays an electric guitar on a city street .
An older man is playing a red guitar on the street .
An older black man plays an electric guitar on the sidewalk .
A man is playing the guitar outdoors in a city .
An older black man plays an electric guitar on a city street .
A white and green helicopter takes off from the airfield .
A helicopter is taking off for flight .
A helicopter about to land on the ground in the middle of clearing in the forest .
A helicopter hovering over a landing spot .
Helicopter taking off from airport .

Fig. 6 Examples of processing cross-modal similarity queries by MCQI

semantic common embedding of two types, but also has explicit alignment information. As shown in Fig. 6, for *k*NN queries, MCQI can return similar objects in datasets and further gives a reason why those objects are semantically related, which is very important for serious applications.

7 Conclusion

In this paper, we proposed a novel framework for Multi-grained Cross-modal Similarity Query with Interpretability (MCQI) to effectively leverage coarse-grained and fine-grained semantic information to achieve effective interpretable cross-modal queries. MCQI integrates deep neural network embedding and high-dimensional query index and also introduces an efficient *k*NN similarity query algorithm with theoretical support. Experimental results on widely used datasets prove the effectiveness of MCQI. In our future work, we will study more reinforcement learning-based cross-modal query approaches for reducing dependence on large training data of certain area.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/s41019-021-00162-4.

Acknowledgements We would like to thank selfless friends and professional reviewers for all the insightful advices. The preliminary version of this article has been published in APWeb-WAIM 2020 [https://doi.org/10.1007/978-3-030-60290-1_26]

Authors' contributions Mingdong Zhu is responsible for providing idea, designing model and experimental methods. Derong Shen gives overall guidance. Lixin Xu implements experiments, Xianfang Wang gives guidance for theory proof.

Funding This work is supported by the National Natural Science Foundation of China (61802116, 62072157), the Training Plan of Young Backbone Teachers in Universities of Henan Province (2020GGJS263), the Natural Science Foundation of Henna province(202300410102), the Science and Technology Plan of Henan Province (192102210113, 192102210248) and the Key Scientific Research Project of Henan Universities (19B520005).

Data availability Open data are used in this work and are publicly available (references are provided in the paper).

Declarations

Conflict of interest Avoid reviewers from Henan Institute of Technology and Northeastern University.

Consent to participate All authors consent to participate in this work.

Consent for publication All authors consent to publish the paper.

References

1. Peng Y, Huang X, Zhao Y (2018) An over view of cross-media retrieval: Concepts, methodologies, benchmarks and challenges. IEEE Trans Circuits Syst Video Technol 28(9):2372–2385
2. He X, Peng Y, Xi L (2019) A new benchmark and approach for fine-grained cross-media retrieval. In: 27th ACM international conference on multimedia, ACM. pp 1740–1748
3. Rasiwasia N, Pereira J, Coviello E et al (2010) A new approach to cross-modal multimedia retrieval. In: 18th international conference on multimedia, ACM. pp 251–260
4. Zhai X, Peng Y, Xiao J (2014) Learning cross-media joint representation with sparse and semisupervised regularization. IEEE Trans Circuits Syst Video Technol 24(6):965–978
5. Peng Y, Zhai X, Zhao Y, Huang X (2016) Semi-supervised cross-media feature learning with unified patch graph regularization. IEEE Trans Circuits Syst Video Technol 26(3):583–596
6. Yan F, Mikolajczyk K (2015) Deep correlation for matching images and text. In: IEEE conference on computer vision and pattern recognition, IEEE. pp 3441–3450
7. He L, Xu X, Lu H et al (2017) Unsupervised cross-modal retrieval through adversarial learning. In: IEEE international conference on multimedia and expo, IEEE. pp 1153–1158
8. Chi J, Peng Y (2020) Zero-shot cross-media embedding learning with dual adversarial distribution network. IEEE Trans Circuits Syst Video Technol 30(4):1173–1187
9. Andrej K, Armand J, Li F (2014) Deep fragment embeddings for bidirectional image sentence mapping. In: 27th international conference on neural information processing systems, ACM. pp 1889–1897
10. Andrej K, Li F (2017) Deep Visual-Semantic Alignments for Generating Image Descriptions. IEEE Trans Pattern Anal Mach Intell 39(4):664–676
11. Xu K, Ba J, Kiros R et al (2015) Show, attend and tell: neural image caption generation with visual attention. In: 2015 international conference on machine learning, IEEE. pp 2048–2057
12. Wang X, Wang Y, Wan W (2018) Watch, listen and describe: globally and locally aligned cross-modal attentions for video captioning. In: Proceedings of 2018 conference of the North American chapter of the association for computational linguistics, ACL. pp 795–801
13. Jiang Q, Li W (2017) Deep cross-modal hashing. In: 2017 IEEE conference on computer vision and pattern recognition, IEEE. pp 3270–3278
14. Cao Y, Long M, Wang J et al (2016) Correlation autoencoder hashing for supervised cross-modal search. In: international conference on multimedia retrieval, ACM. pp 197–204
15. Cao Y, Long M, Wang J (2017) Correlation hashing network for efficient cross-modal retrieval. In: 28th British machine vision conference, BMVA. pp 1–12
16. Yang E, Deng C, Liu W et al (2017) Pairwise relationship guided deep hashing for cross-modal retrieval. In: 31st conference on artificial intelligence, AAAI. pp 1618–1625
17. Zhang J, Peng Y, Yuan M et al (2018) Unsupervised generative adversarial cross-modal hashing. In 32nd conference on artificial intelligence, AAAI. pp 539–546
18. Yang K, Ding X, Zhang Y et al (2019) Distributed similarity queries in metric spaces. Data Science and Engineering 4(4):1–16
19. Batko M (2004) Distributed and scalable similarity searching in metric spaces. In: 9th EDBT, ACM. pp 44–153
20. Novak D, Batko M (2011) Zezula P, Metric index: An efficient and scalable solution for precise and approximate similarity search. Inf Syst 36(4):721–733
21. Wang J, Wu S, Gao H et al (2010) Indexing multi-dimensional data in a cloud system. In: SIGMOD, ACM. pp 591–602
22. Wu S, Jiang D, Ooi B, Wu K (2010) Efficient B-tree based indexing for cloud data processing. In: 36th VLDB, ACM. pp 1207–1218
23. Tanin E, Harwood A, Samet H (2007) Using a distributed quadtree index in peer-to-peer networks. VLDB J 16(2):165–178
24. Bennanismires K, Musat C, Hossmann A et al (2018) Simple Unsupervised Keyphrase Extraction using Sentence Embeddings. In: conference on computational natural language learning, ACL. pp 221–229
25. Shen Y, He X, Gao, J et al (2014) A latent semantic model with convolutional-pooling structure for information retrieval. In: conference on information and knowledge management, ACM. pp 101–110
26. Cheng B, Wei Y, Shi H et al (2018) Revisiting RCNN: On awakening the classification power of faster RCNN. In: European conference on computer vision, Springer. pp 473–490
27. Cer D, Yang Y, Kong S et al (2018) Universal Sentence Encoder. arXiv: Computation and Language. https://arxiv.org/abs/1803.11175v2. Accessed 12 April 2018
28. Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: 13th international conference on artificial intelligence and statistics, JMLR. pp 249–256
29. Zhu M, Xu L, Shen D et al (2018) Methods for similarity query on uncertain data with cosine similarity constraints. Journal of Frontiers of Computer Science and Technology 12(1):49–64
30. Hodosh M, Young P, Hockenmaier J (2013) Framing image description as a ranking task: data, models and evaluation metrics. Journal of Artificial Intelligence Research 47(1):853–899
31. Young P, Lai A, Hodosh M et al (2014) From image descriptions to visual denotations: New similarity metrics for semantic inference over event descriptions. Transactions of the Association for Computational Linguistics 7(2):67–78
32. Chua T, Tan J, Hong R et al (2009) NUS-WIDE: a real-world web image database from national university of Singapore. In: 8th conference on image and video retrieval, ACM. pp 1–9
33. Lin T, Maire M, Belongie S (2014) Microsoft coco: Common objects in context. In: 13th European conference on Computer Vision (ECCV), Springer. pp 740–755
34. Peng Y, Qi J, Huang X et al (2018) CCL: Cross-modal correlation learning with multigrained fusion by hierarchical network. IEEE Trans Multimedia 20(2):405–420
35. Chen T, Wu W, Gao Y et al (2018) Fine-grained representation learning and recognition by exploiting hierarchical semantic embedding. In: 26th ACM multimedia, ACM. pp 2023–2031
36. Lee K, Chen X, Hua G et al (2018) Stacked cross attention for image-text matching. In: European conference on computer vision, Springer. pp 212–228
37. Kang C, Xiang S, Liao S et al (2015) Learning Consistent Feature Representation for Cross-Modal Multimedia Retrieval. IEEE Trans Multimedia 17(3):370–381

38. Hardoon D, Szedmak S, Shawetaylor J et al (2004) Canonical correlation analysis: An overview with application to learning methods. Neural Comput 16(12):2639–2664
39. Akdogan A, Demiryurek U, Kashani FB et al (2010) Voronoi-based geospatial query processing with mapreduce. In: 2nd international conference of cloud Computing(CloudCom), IEEE. pp 9–16
40. Abadi M, Barham, P, Chen J et al (2016) TensorFlow: A system for large-scale machine learning. In: 12th USENIX conference on operating systems design and implementation, ACM. pp 265–283