Future Generation Computer Systems 102 (2020) 53–65

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

Semantic-aware data quality assessment for image big data

Yu Liu, Yangtao Wang, Ke Zhou [*], Yujuan Yang, Yifei Liu

Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China

HIGHLIGHTS

- Data quality assessment is essential for realizing the promise of big data.
- Relevance can arouse the user's interest in exploiting the data source.
- IDSTH algorithm can extract semantic features with generalization ability.
- SHR algorithm calculates the importance score (rank) for each node (image).
- SDQA architecture can help assess the value of image big data.

ARTICLE INFO

Article history:
Received 28 January 2019
Received in revised form 11 June 2019
Accepted 25 July 2019
Available online 6 August 2019

Keywords:
Semantic-aware
Quality assessment
Image big data
IDSTH
SHR

ABSTRACT

Data quality (DQ) assessment is essential for realizing the promise of big data by judging the value of data in advance. Relevance, an indispensable dimension of DQ, focusing on "fitness for requirement", can arouse the user's interest in exploiting the data source. It has two-level evaluations: (1) the amount of data that meets the user's requirements; (2) the matching degree of these relevant data. However, there lack works of DQ assessment at dimension of relevance, especially for unstructured image data which focus on semantic similarity. When we try to evaluate semantic relevance between an image data source and a query (requirement), there are three challenges: (1) how to extract semantic information with generalization ability for all image data? (2) how to quantify relevance by fusing the quantity of relevant data and the degree of similarity comprehensively? (3) how to improve assessing efficiency of relevance in a big data scenario by design of an effective architecture?

To overcome these challenges, we propose a semantic-aware data quality assessment (SDQA) architecture which includes off-line analysis and on-line assessment. In off-line analysis, for an image data source, we first transform all images into hash codes using our improved Deep Self-taught Hashing (IDSTH) algorithm which can extract semantic features with generalization ability, then construct a graph using hash codes and restricted Hamming distance, next use our designed Semantic Hash Ranking (SHR) algorithm to calculate the importance score (rank) for each node (image), which takes both the quantity of relevant images and the degree of semantic similarity into consideration, and finally rank all images in descending order of score. During on-line assessment, we first convert the user's query into hash codes using IDSTH model, then retrieve matched images to collate their importance scores, and finally help the user determine whether the image data source is fit for his requirement. The results on public dataset and real-world dataset show effectiveness, superiority and on-line efficiency of our SDQA architecture.

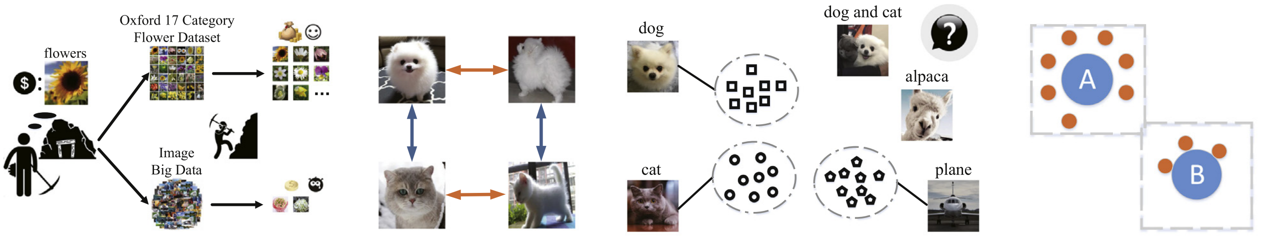
1. Introduction

With social platforms rapidly developing, the amount of image data has been growing exponentially. However, most of image data stored in the device fail to play their value, because data consumers who lack awareness of their contents, dare not audaciously use them. The reason for "lack of awareness" is not only the lack of labels and associations in storage, but also the limitations on the processing of unstructured data. Especially in a big data scenario, the implementation of data cleaning approaches is not feasible due to the size and the streaming nature of the data source, making the use of image big data more difficult [1].

Data quality assessment is a key to realize the promise of big data by judging the value of data in advance. Data Quality (DQ), generally defined as "fitness for use", is evaluated by five dimensions that include availability, reliability, relevance, usability and presentation quality [2]. Availability and reliability have been well studied in big data integration and fusion [3], where availability is defined as the degree of convenience for data accessing

[*] Corresponding author.
E-mail addresses: liu_yu@hust.edu.cn (Y. Liu), ytwbruce@hust.edu.cn (Y. Wang), k.zhou@hust.edu.cn (K. Zhou), gracee@hust.edu.cn (Y. Yang), yifeiliu@hust.edu.cn (Y. Liu).

https://doi.org/10.1016/j.future.2019.07.063
0167-739X/© 2019 Published by Elsevier B.V.



(a) mining and wealth (b) semantic-aware and data-aware (c) out-of-sample problem in classification (d) evaluation dilemma

Fig. 1. (a) Mining does not mean wealth, because the data source may not be relevant. However, what if the user can perceive the relevance in advance? (b) The orange arrows represent the semantic-aware results, while the blue arrows represent the data-aware results. (c) The model is hard to classify "dog and cat" and "alpaca". (d) The orange nodes represent the data source. The blue nodes represent the user's query. The distances between different nodes represent the degree of similarity. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

(i.e. accessibility, timeliness and authorization), while reliability refers to whether the user can trust the data (i.e. accuracy, integrity, consistency, completeness and auditability). However, neither of them can help the user determine whether the data source meets his demands. Paying all attention to availability and reliability is not enough to arouse the user's interest in exploiting the data source.

Relevance, as another indispensable dimension of DQ, is more crucial to judge "fitness for requirements". It is defined as the degree of correlation between the content of data and the user's expectations or demands. There are two-level evaluations for the relevance: (1) the amount of data that meets the user's requirements; (2) the matching degree of these relevant data. However, it is tricky to achieve these evaluations, especially for unstructured image data which focus on semantic similarity. Previous attempts at image data quality assessment pay attention to evaluate the quality of image presentation or retrieval results, which is inconsistent with our topic. Besides, existing DQ assessment jobs prefer to adopt data-aware methods to achieve evaluation at other dimensions. As a result, there lacks jobs evaluating relevance by semantic-aware methods, especially for image big data. On the other hand, for similarity comparison, semantic-aware feature extraction methods have been well studied in image retrieval, but those techniques have not been integrated into DQ assessment.

In this paper, we propose a semantic-aware data quality assessment (SDQA) architecture for image big data, where we design our improved deep self-taught hashing (IDSTH) algorithm, semantic hash ranking (SHR) algorithm and semantic-aware assessment process. Compared with the traditional DQ assessment, we focus on the relevance and bring three improvements: (1) we extract semantic feature with generalization ability and complete hash mapping for large-scale unlabeled image data; (2) we evaluate and quantify the relevance based on both the quantity of relevant images and the degree of semantic similarity; (3) we introduce deep learning and image retrieval technologies into the architecture, and combine off-line analysis with on-line evaluation to complete relevance assessment. Based on this, we ensure the efficiency of assessment in large-scale scenarios. To evaluate our work, we implement our SDQA on public datasets with classification supervision information. Our results show SDQA is sensitive to semantic content and gives higher scores to images whose semantic contents account for higher proportion in the whole dataset. At last, we show the relevance assessment result on real-world datasets for given requirements.

2. Motivation

In practical applications, data mining is like a gamble. There is a high chance that we have spent huge mining cost but fail to get corresponding value. This is not because the data mining algorithms are not good enough, but because the data cannot necessarily bring value to the demands (applications). As shown in Fig. 1(a), faced with the demand of "searching flowers", Oxford 17 Category Flower Dataset could provide enough value but Image Big Data failed in this aspect. This inherent value relationship between the data and demands can never be changed by data mining algorithms, but the degree of this relationship can be judged and measured in advance to avoid the gamble behavior as much as possible.

For judging and measuring this kind of relationship, it requires a unified parsing mechanism to extract both the semantic information of data and demands. However, semantic extraction for unstructured image data is restricted by the generalization ability and thus researchers doubt about the practicality of this conduct in a big data scenario. As a result, existing DQ assessment lacks exploration in the relevance of data. When we try to evaluate semantic relevance between an image data source and a query (requirement), there are three challenges: (1) how to extract semantic information with generalization ability for all image data? (2) how to quantify relevance by fusing the quantity of relevant data and the degree of similarity comprehensively? (3) how to improve assessing efficiency of relevance in a big data scenario by design of an effective architecture? We now discuss these issues in further detail.

(1) Semantic feature with generalization ability: semantic features are data representations that express human cognition. Different from data-aware feature extraction (commonly used in conventional DQ assessment) which entirely relies on the data distribution itself, semantic-aware methods rely more on hand-crafted labels and their results are suppose to be more meaningful. As shown in Fig. 1(b), according to what the blue arrows connect, dogs and cats have the same pixel distribution, so data-aware methods consider them to be the same category. However, dogs and cats have completely different semantic contents, as connected by the orange arrows. This is because data-aware methods care about what the data (pixel) looks like, while semantic-aware ones focus on what the data itself is.

Semantic-aware feature extraction has been well studied in image classification, which can ensure the distance between same categories data is getting smaller and smaller while the one between different categories is getting bigger and bigger. However, this way loses the generalization ability to cognize objects, which leads to serious out-of-sample problems. As shown in Fig. 1(c), the model that has learned single dog, cat and plane can classify images of single cat, dog and plane well, but it fails to classify "dog and cat" and recognize "alpaca". It is impossible that a model can learn all entities in real world, so semantic feature extraction methods without generalization ability cannot apply to big data scenarios.

We believe what causes this defect is that classification methods ignore the meaning of the differences (distances) between different categories. For example, cats as animals are more similar to dogs (which also belong to animals) than airplanes (which belong to machines). This apparent difference is reflected by the data itself, which is the principle followed by data-aware feature extraction methods. Therefore, we are supposed to combine both semantic-aware and data-aware advantages to learn the classification according to “what it is” and the distance between categories according to “what it looks like”. Based on this, semantic-aware methods will be improved to own generalization ability and practicality.

(2) Relevance evaluation method: there lacks assessment work about relevance (i.e. the quantity of relevant image and degree of semantic similarity). In practice, we find it difficult to measure these two evaluations. First, it is impossible to get them in isolation, because a convincing threshold for quantity or degree is hard to define. Second, even if respectively getting them, it is tricky to judge which is more representative of relevance. As shown in Fig. 1(d), “A” has more relevant data while “B” is closer (shorter distances). However, which one is more relevant to the image big data? Therefore, it is necessary to evaluate the relevance fusing the two evaluations.

Taking both evaluations into consideration comprehensively, clustering may be a good method to evaluate each data. However, whether the method is based on the number of hypothetical centers [4] or density [5], it needs many iterations and will take a long time to stabilize. This overhead is unacceptable for big data, albeit in the off-line manner. Also, quantization learning method [6] uses the concept of codebook to specify the evaluation standard in clustering, but codebook is only suitable for encoding data and thus fails to give the overall metric. Beyond that, graph-based computing [7], especially PageRank algorithm [8], is a wonderful choice to achieve global evaluation, whereas more factors need to be taken into account. If we can use random walk on undirected graph whose edges’ weights denote semantic distances between nodes, the evaluation will be completed considering both the quantity of relevant images and the degree of semantic similarity.

(3) Architecture with efficiency: as a rule of thumb, an assessment framework at least includes the user’s query as input, data source analysis and conclusion as output. Because of acquirement for semantic information and comparison for query, deep learning and retrieval mechanism are integrated into our assessment framework including data source training, semantic extraction and vector matching computing. Therefore, it is an exacting task to design a reasonable architecture that meets the efficiency of large-scale scenarios.

In terms of deep learning, because the scope of search is the data source, it means that semantic extraction model must be trained on the data source for perceiving its semantic distribution. Moreover, the content of data source is relatively fixed. Therefore, the model training and feature extraction can be off-line completed. Note that semantic features of the query must be extracted by above model. As for matching of retrieval, to avoid comparison of millions of high-dimensional floating-point vectors which brings long time latency, we are supposed to build more efficient measurement, e.g., hashing techniques.

In addition, for requirements that retrieve images, the way that users directly input images may be simpler and more intuitive. Similarly, the output is supposed to be digital values that intuitively express the semantic relevance.

3. SDQA design

We now present our SDQA design that overcomes the challenges above.

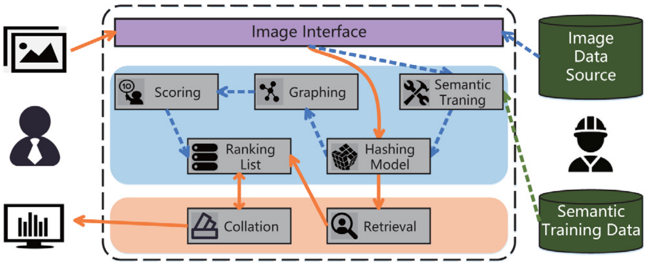


Fig. 2. Semantic-aware DQ assessment architecture. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

3.1. Problem formulation

Given an image data source and a query image (or a query with multiple images), we hope to (1) learn a model for semantic extraction with generalization ability; (2) find the matched images corresponding to the query; and (3) return a score reflecting the relevance of the query to the image data source.

For case (1), assuming that the extracted semantic features sets of cat, dog and airplane are respectively denoted as C, D and P, X_i and Y_j represent different features that respectively belong to X and Y, where $X, Y \in \{C, D, P\}$. $\mathcal{FD}(X_i, Y_j)$ represents the distance between X_i and Y_j. First, we require $\mathcal{FD}(X_i, X_j) < \mathcal{FD}(X_i, Y_j)$, where $X \neq Y$. Beyond that, we require $\mathcal{FD}(C_i, D_j) < \mathcal{FD}(C_i, P_j)$ and $\mathcal{FD}(C_i, D_j) < \mathcal{FD}(D_j, P_i)$ to ensure the similarity (“what it looks like”) between closer categories under the condition of accurate classification (“what it is”).

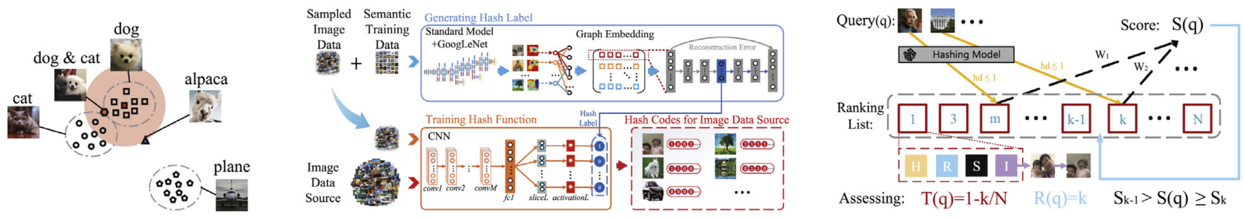
For case (2), we need to define whether two images are matched. To this end, we compute the hash code for each image, and two images are matched (or similar) if their Hamming distance is not larger than a given threshold. Formally, assuming that h_i and h_j respectively represent the hash codes of two images, we use $\mathcal{HD}(h_i, h_j)$ to denote the Hamming distance between h_i and h_j, and hd to denote a given matching threshold. If $\mathcal{HD}(h_i, h_j) \leq hd$, the two images are matched.

For case (3), we evaluate the relevance to the query. To this end, we rank all the N images (contained in the data source) based on their semantics, and denote the ranked scores as $\{S_1, S_2, \dots, S_N\}(\forall k, S_{k-1} \geq S_k)$. Then given a query image, we find all the matched images and calculate the average weighted score of all these matched images, and denote this score as $S(q)$. If $S_{k-1} > S(q) \geq S_k$, we then return the ratio $T(q) = 1 - \frac{k}{N}$ as the query score, which shows how much this query is related to the data source. Obviously, the larger $T(q)$, the higher relevance of the query to the data source.

3.2. Architecture overview

SDQA is an architecture that works to assess data quality of the image data source by semantic relevance. As shown in Fig. 2, the architecture contains an off-line (light blue background) part and an on-line (light orange background) part. The purpose of this design is to reduce the on-line workload and ensure the real-time efficiency.

Off-line part: off-line part includes modules of *semantic training, hashing model, graphing, scoring* and *ranking list*. The purpose of this part is to get the semantic hashing model and the data ranking list according to the semantic distribution of image data source, which provides the resource for the on-line part. Note that *semantic training* and *hashing model* together constitute the IDSTH model, which overcomes the challenge (1) in Section 2. Module



(a) semantic distances (b) improved deep self-taught hashing (c) ranking list and collation

Fig. 3. (a) Results of feature extraction with generalization ability. (b) The framework of improved DSTH. (c) The processing of retrieval and collation on ranking list.

of *scoring* is completed by SHR algorithm, which overcomes the challenge (2) in Section 2.

It is worth noting that DSTH [9,10] is a published work, which introduces Laplacian matrix into the deep model to re-divide the distance between data from a global perspective, so that the deep model can learn the inherent mechanism of this global division. With the development of graph embedding technique, Laplacian matrix decomposition has been proved to only reflect the first-order proximity between data, while the second-order proximity with stronger relationship expression ability needs to be realized by learning the adjacency matrix. Therefore, in our deep model, we adopt adjacency matrix instead of Laplacian matrix to improve both the way of generating hash label and the mechanism of dividing distance, in order to obtain stronger generalization ability of IDSTH.

On-line part: on-line part includes modules of *retrieval* and *collation*. The purpose of this part is to quickly retrieves one or more matched images from data source according to the user's query images, and calculates the ranking (ration) to provide users with relevance judgment. Note that *retrieval* imposes comparison of Hamming distances to ensure the efficiency and *collation* returns the ranking (ratio) to express relevance, which overcome the challenge (3) in Section 2.

In addition, the *image interface* is designed to access images and convert them into 256 × 256 RGB format. We now introduce above modules respectively.

3.3. Semantic training

The purpose of this module is to perceive the semantic distribution of the data source and generate hash labels. Fusing the characteristics of "what it is" and "what it looks like", *semantic training* learns the semantic distance and maps all data to hash codes. As shown in the blue frame (generating hash label) of Fig. 3(b), we use both semantic training data and image data sampled from image data source as the input, and achieve the feature extraction on trained GoogLeNet. Then we construct a graph using the Euclidean distance of those features to obtain the adjacency matrix. Finally, we input each row of the matrix to the AutoEncoder model. With the minimum reconstruction error, we use the intermediate encoding results (dark blue layer) for binarization to acquire hash labels. The detailed processing is introduced in Section 4.

There are two points that we must explain. First, semantic training data refer to those data that have achieved good classification results on the standard GoogLeNet model. We use them together with sampled image data as input, aiming at in the error reconstruction stage, restricting the semantic bias of extracted features caused by the standard model which has not perceived the semantic distribution of the data source, so as to ensure the quality of "what it is". Second, transforming a graph into hash labels is actually a deep graph embedding process, which can express and learn the distance for all data in the data distribution itself, so that "what it looks like" can be well reflected. As shown in Fig. 3(a), dog and cat looks similar, so they have closest semantic distance. Besides, although alpaca looks unlike cat or dog, it also belongs to animals, and thus has closer semantic distance to dog and cat. However, plane which belongs to machines, has completely different semantic content from animals, so it has the longest semantic distance from dog. In addition, compared with DSTH which empirically sets the number of neighbors, our improved method can automatically and objectively determine the neighbors. At the same time, IDSTH does not have to rely on the whole large adjacency matrix, but can independently complete fitting for each row, which provides a more flexible choice for time-memory trade-off.

3.4. Hashing model

The purpose of this module is to learn a hash model for quickly semantic extraction and mapping. Acquirement of hash codes in semantic training module has to wait for the calculation of distances among all images, which is unpractical for big data real-time scenes. Therefore, we learn a hash function by fitting above hash codes as labels using convolution neural network, as shown in the orange frame (training hash function) in Fig. 3(b). We introduce the detailed training process in Section 4. Note that in the hash function training stage, we only need to train the sampled image data. The semantic expression of both the data source and query is accomplished by this module.

3.5. Graphing

The purpose of graphing is to associate all image data using semantic distance. We model all images as a graph G where each node is an image and edges are relationships between images. In order to speed up the graph construction, we cut off those edges on which the weight exceeds half of the length of hash code, according to the conclusion of Long [11]. Let N_* denote the $*$-th node of G, $H(N_*)$ denote hash code of N_* and l denote length of hash codes. We define XOR operation as $\oplus$. Therefore, the Hamming distance weight on the undirected link between N_i and N_j can be defined as

$$d_{ij} = \begin{cases} H(N_i) \oplus H(N_j) & i \neq j, H(N_i) \oplus H(N_j) \leq \Omega, \\ NULL & otherwise. \end{cases} \tag{1}$$

where $\Omega = \lceil l/s \rceil$ and $s \in [1, l]$. In practice, the determination of Ω is based on efficiency of building a graph with tolerable loss. Formally, we define the precision of ith node as C_i/L_i, where L_i represents the number of all nodes connected to ith node and there exist C_i nodes of the L_i nodes that have the same label as the ith node. Therefore, the precision of graph $P(G|\Omega)$ is defined as

$$P(G|\Omega) = \frac{1}{N} \sum_{i=1}^{N} \frac{C_i}{L_i} \tag{2}$$

3.6. Scoring

After building the graph with restricted Hamming distance, we calculate the importance score for each node by random walk. We extend the PageRank algorithm and propose the SHR algorithm which takes both the number of connected links and the weight on edges into consideration. Note that we specially design how to reasonably calculate the extent of relevance between nodes, aiming at making full use of the Hamming distance of similarity hash with generalization ability. As shown in Fig. 3(a), the red dot contained in the dog category is the query data, and the shaded part centered with this dot represents those data directly connected with it. Because the data that looks similar is closer to each other, images whose partial content contains dog will be connected to this shaded part. Besides, the high-level semantic information of dog, i.e. animal, has been better reflected.

On the graph, we use SHR to calculate the importance score for each node. Specifically, according to the physical meaning of Hamming distance, we redesign the iteration matrix elements for obtaining reasonable importance scores. SHR makes the dominant semantics more prominent, thus reinforcing the user's cognition of the date source. We introduce the detailed calculation process in Section 5.

3.7. Ranking list

The purpose of *ranking list* is to provide look-up table for online assessment. We rank all the images in a descending order of their importance scores. As shown in Fig. 3(c), each unit in ranking list contains four elements: $\mathcal{H}$ denotes hash code, $\mathcal{R}$ denotes rank, $\mathcal{S}$ denotes score, and $\mathcal{I}$ denotes the images presented by $\mathcal{H}$. We set that $\mathcal{H}$ is the key, because multiple images may share a same hash code and SHR gives the same hash code the same score.

3.8. Retrieval and collation

According to the Hamming distance, hash codes of the user's query (images) retrieve suitable images contained in the data source. We collect their scores and collate them with different weights. As shown in Fig. 3(c), for the ranking list consisting of N images, the query is mapped to hash codes by the hashing model and retrieves images. The matching range is defined as hd and we set $hd = 1$. Mathematically, we let q denote a query with n images, img_i denote the ith image where $i \in [1, n]$, m_i denote the number of matched images for the ith image of the query q. Meanwhile, we let $S_j(img_i)$ denote the score of the jth image where $j \in [1, m_i]$. Therefore, the score of q is defined as follows:

$$S(q) = \sum_{i=1}^{n} \frac{1}{m_i} \sum_{j=1}^{m_i} \beta_i S_j(img_i) \qquad s.t. \sum_{i=1}^{n} \beta_i = 1 \tag{3}$$

where $\beta_i \in [0, 1]$ represents the importance weight of the ith image.

Compared with the scores denoted as $\{S_1, S_2, \ldots, S_N\}$ of image data source, we can acquire the rank of $S(q)$ denoted as k, where $S_{k-1} > S(q) \geq S_k$. Further, $T(q) = 1 - \frac{k}{N}$ represents relevance of image data source to the query.

3.9. Example

As shown in Fig. 2, the assessment process includes off-line analysis (blue dotted line) and on-line assessment (orange solid line). In practice, we first carry out off-line analysis once and then repeatedly implement on-line assessment. We respectively denote an image data source and a query as *IDS* and q.

Off-line analysis consists of five steps. (1) In the *semantic training* module, the administrator prepares semantic training data (e.g. ImageNet data), feature extraction network (e.g. GoogLeNet inception V4), feature extraction location (e.g. the last pooling layer), AutoEncoder network and its output dimension (e.g. 48-bits). A certain proportion of data (e.g. 60%) are sampled from *IDS*. After formatted by image interface, those sampled image data (3 × 255 × 255 dimension) together with the semantic training data are input to the feature extraction network. Then, we obtain the feature vectors (e.g. 4096-dimension) to complete the training of graph embedding and acquire the hash codes of sampled image data. (2) In the *hashing model* module, we train the sampled image data using their corresponding hash codes as labels via convolutional neural network(CNN). Then, we translate all *IDS* into hash codes by trained hashing model. (3) In *graphing* module, we construct a graph according to the restricted Hamming distances between images, and compute its adjacent matrix. (4) In *score* module, according to above matrix, we use SHR algorithm to iteratively calculate the importance score for each node. (5) In *ranking list* module, hash codes are used as the key to construct hash table. According to the importance scores, we record and arrange each hash code including its ranking, score and corresponding images.

During on-line assessment, users input query images which express their requirements. For example, if one desires the resource related to Formula 1, he is supposed to input images including car, racetrack and racing drivers, etc. After formatted by image interface, these query images are translated into hash codes using trained hashing model. According to those hash codes, we collect matched images (hash codes and scores) within given Hamming distance range ($hd \leq 1$) from the *ranking list*, and then use *collation* module to calculate the average weighted scores of these matched images to get the semantic score of query images. At last, we resort the ranking list again to determine the importance ranking of the semantic score in *IDS*, and return $T(q)$ to users as relevance judgment.

4. Improved Deep Self-taught Hashing (IDSTH)

In this section, we detailedly describe the design of IDSTH algorithm including stage of hash label generating (how to learn semantic features extraction for data source without labels, how to fuse the semantic information with categories distances into deep learning framework and how to generate hash labels) and stage of hash function training (how to conduct the training of hashing model).

4.1. Hash label generating

The prime task of this stage is to perceive data distribution of the data source and map the semantic information into hash codes as labels for the next stage. The labels determine which semantic informations should be extracted from data in the subsequent function learning. Therefore, the hash codes are supposed to own generalization ability mentioned in Section 2. Supervised deep learning algorithm is able to better extract semantic information owing to the accurate hand-crafted labels, while structural deep network embedding algorithm can express the relationships between data, and map them to the compact space.

We combine these two methods, which can not only acquire semantic information without labels, but also reach the balance between human semantic cognition and data distribution itself, so as to acquire our expected semantic hash codes (labels). The whole processing is shown in the blue frame of Fig. 3(b).

Given this, in the absence of labels, we use the standard deep model to get features for both image data sampled from the data source and semantic training data which have been trained by the standard deep model. Note that there will inevitably produce semantic bias if using the standard deep model to extract sampled image data, because this model is not trained on the data source and thus loses cognition about its semantic distribution. (In practice, it is impossible to use the standard model to fine-tune the data source which have no labels.) Therefore, we utilize the semantic features extracted from the training data on standard model, constrain those features with semantic bias via associated relationships, and correct this bias by the experience (capturing the topological structure of a graph) of fitting accurate data during the mapping process. In the implementation, we choose structural deep network embedding algorithm to complete this process.

Mathematically, according to feature extraction for sampled image data and semantic training data, we acquire m z-dimensional feature vectors $\mathcal{FV} = \{v_1, v_2, \dots, v_m\}$ where $\forall v \in \mathbb{R}^z$. Meanwhile, we construct a graph $\mathcal{G} = (\mathcal{FV}, \mathcal{SE})$, where $\mathcal{FV}$ represents m vertexes and $\mathcal{SE} = \{e_{i,j}\}_{i,j=1}^m$ represents the edges, where $i, j \in \mathbb{Z}^+$. Each edge $e_{i,j}$ is associated with a weight $u_{i,j}$ defined as

$$u_{i,j} = \begin{cases} \dfrac{\vec{v}_i \cdot \vec{v}_j}{\|v_i\| \cdot \|v_j\|} & i \neq j, \\ 0 & i = j. \end{cases} \tag{4}$$

We obtain $\mathcal{G}$'s adjacency matrix U which consists of m elements $u_1, u_2, \dots, u_m$. Each element $u_i = \{u_{i,j}\}_{j=1}^m$ provides the semantic distances distribution of each image. Leaning the latent presentation of all data from the adjacency matrix can accurately reflect the differences between different categories data, so that the distances between categories can be all reflected. With U, we use AutoEncoder model to map this structure and preserve the structural proximity.

AutoEncoder is a kind of unsupervised learning method, which includes the encoder and the decoder. The encoder maps the input data to the compact space through multiple layers containing non-linear functions, while the decoder maps the result of compact space to reconstruction space by the same way. Then given the input u_i, the output for each layer are shown as follows:

$$f_i^{(k)} = \begin{cases} \varphi(W^{(k)} u_i + b^{(k)}) & k = 1, \\ \varphi(W^{(k)} f_i^{(k-1)} + b^{(k)}) & k = 2, \dots, K. \end{cases} \tag{5}$$

where $f_i^{(k)}$, $W^{(k)}$ and $b^{(k)}$ denote output, weight and bias of kth layer respectively. $\varphi(x) = \frac{1}{1+exp(-x)}$ (*sigmoid* function) is the activation function. When $k = K$, we can acquire g-dimensional $\widetilde{u}_i$ as the result of encoder. Then we can acquire the output $\widehat{u}_i$ as the result of reconstruction space by reversing the above processing. The loss function for minimum of reconstruction error is shown as follows:

$$\mathcal{L}_e = \sum_{i=1}^{m} \|u_i - \widehat{u}_i\|_2^2 \tag{6}$$

The reconstruction criterion can smoothly capture the data manifolds and preserve the similarity between data. In addition, using the adjacency matrix to learn the second-order (global) proximity to obtain the distance between different categories data, we also hope to learn the first-order (local) proximity of connected nodes (that DSTH achieved by Laplacian Eigenmaps) to ensure the classification accuracy of the same categories data. The loss function for this goal is defined as follows:

$$\mathcal{L}_d = \sum_{i,j=1}^{m} u_{i,j} \|\widetilde{u}_i - \widetilde{u}_j\|_2^2 \tag{7}$$

Compared with the Laplacian Eigenmaps method, $\mathcal{L}_d$ can control the mapping result more intuitively by the distance. Finally, in order to prevent over-fitting, we design the regularizer term as follows:

$$\mathcal{L}_r = \frac{1}{2} \sum_{k=1}^{K} \|W^{(k)}\|_F^2 + \|\widehat{W^{(k)}}\|_F^2 \tag{8}$$

Based on above design, the terminal loss function $\mathcal{L}$ is:

$$\mathcal{L} = \mathcal{L}_e + \alpha \mathcal{L}_d + \beta \mathcal{L}_r \tag{9}$$

where α and β are hyper-parameters.

When $\mathcal{L}$ reaches convergence, we convert the g-dimensional real-valued vector $\widetilde{u}_1, \widetilde{u}_2, \dots, \widetilde{u}_m$ into binary codes according to the threshold. We set ϱ^p and $\bar{u}_i^p$ to denote threshold and element of pth bit of $\widetilde{u}_i$. The hash label as final result value of $\bar{u}_i^p$ is:

$$\bar{u}_i^p = \begin{cases} 1 & \widetilde{u}_i^p \geqslant \varrho^p, \\ 0 & otherwise. \end{cases} \tag{10}$$

where $1 \leq p \leq g$ and

$$\varrho^p = \frac{1}{m} \sum_{i=1}^{m} \widetilde{u}_i^p \tag{11}$$

4.2. Hash function training

The primary work of this stage is to learn and get the hash function model based on the hash labels obtained from the first step. The reason why we need to re-learn these hash codes is that the graphing process in the first step severely affects the efficiency of generating hash codes, and we hope the hash model applied to on-line process is end-to-end.

We employ convolutional neural network (CNN) to receive fine-grained features. After that, we adopt encoding module which is Divide and Encode Module [12] associated with activation function of BatchNorm [13] to approximate hash labels generated in previous stage. The learning framework is the artificial neural network on the multi-output condition. The whole processing is shown in the orange frame of Fig. 3(b). Formally, we set a function $f : \mathbb{R}^I \to \mathbb{R}^O$, where I is the input set, O is the output set and x is the input vector (image).

$$\begin{aligned} &f^{(1)}(x) = \sigma(W^{(1)} x + b^{(1)}) \\ &f^{(k)}(x) = \sigma(W^{(k)} f^{(k-1)}(x) + b^{(k)}) \end{aligned} \tag{12}$$

where $k = 2, 3, \dots, Q$, Q is number of layers and $\sigma(\cdot)$ is ReLU and BatchNorm function. When the core of $\sigma(x)$ is BatchNorm, the function is calculated as follows:

$$\tilde{x}^{(k)} = \frac{x^{(k)} - E\left(x^{(k)}\right)}{\sqrt{Var\left(x^{(k)}\right)}} \tag{13}$$

where

$$E(x) = \frac{1}{m} \sum_{i=1}^{m} x_i \tag{14}$$

$$Var(x) = \frac{1}{m} \sum_{i=1}^{m} (x_i - E(x))^2 \tag{15}$$

In the last layer of CNN, we split a 1024-dimensional vector into 8 groups, and each group is mapped to $g/8$ elements to fit the g-dimensional label. We use $\bar{x}_i$ to denote the output vector. The loss function $\mathcal{L}_h$ is:

$$\mathcal{L}_h = \sum_{i=1}^{m} \|\bar{x}_i - \bar{u}_i\|_2^2 \tag{16}$$

At last, we apply the threshold values of each bit calculated in the hash label generating stage.

5. Semantic hash ranking (SHR)

In this section, we introduce SHR algorithm in detail. SHR considers both the number of connected links and the weight on edges into consideration, reasonably designs impact factor between different nodes according to similarity hash code, and calculate the importance score for each node by random walk. We also give a concise description of its convergence.

Let L_* denote number of links to N_*. Draw rank factor $R(N_*)$ for N_* and impact factor $I(N_{ij})$ for N_j to N_i, where $I(N_{ij})$ is defined as

$$I(N_{ij}) = \begin{cases} \dfrac{l - d_{ij}}{\sum_{t\in T_j} l - d_{tj}} R(N_j) & \exists d_{ij}, \\ 0 & otherwise. \end{cases} \tag{17}$$

where T_j is the set including orders of all nodes associated with N_j. Specially, we design the formulation according to two principals. Firstly, the less d_{ij} is, the greater influence N_j contributes to N_i is. Meanwhile, the longer hash code is, the more compact the similarity presented by d_{ij} is. Secondly, PageRank considers the weights on each edge as the same, but we extend it to be applied to different weights on edges. As a result, when weights on different edges are the same, Eq. (17) should be the same as the impact factor formulation of PageRank. Consequently, $R(N_i)$ should be equal to the sum of the impact factors of all nodes linked to N_i

$$R(N_i) = \sum_{j=1, j\neq i}^{n} I(N_{ij}) \tag{18}$$

Let f_{ij} represent the coefficient of $R(N_j)$ in $I(N_{ij})$. We draw iteration formula as

$$\begin{bmatrix} R^{c+1}(N_1) \\ R^{c+1}(N_2) \\ \cdots \\ R^{c+1}(N_n) \end{bmatrix} = \begin{bmatrix} 0 & f_{12} & \cdots & f_{1n} \\ f_{21} & 0 & \cdots & f_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ f_{n1} & f_{n2} & \cdots & 0 \end{bmatrix} \begin{bmatrix} R^{c}(N_1) \\ R^{c}(N_2) \\ \cdots \\ R^{c}(N_n) \end{bmatrix} \tag{19}$$

where c is the number of iteration rounds. We define the termination condition as

$$R^{c+1}(N_m) - R^{c}(N_m) \leq \varepsilon \tag{20}$$

where $m \in [1, n]$ and $\forall N_m$ should satisfy Eq. (20). Meanwhile, ε is constant. Let $SHR(N_*)$ denote semantic rank of N_*. The last results are

$$SHR(N_m) = R^{\eta}(N_m) \tag{21}$$

where η is the round on termination.

5.1. Convergence

In order to ensure our algorithm can converge to a stable result, we prove the convergence of Eq. (19), let A_n represent iteration coefficient matrix. The A_n is

$$A_n = \begin{bmatrix} 0 & f_{12} & \cdots & f_{1n} \\ f_{21} & 0 & \cdots & f_{2n} \\ \cdots & \cdots & \cdots & \cdots \\ f_{n1} & f_{n2} & \cdots & 0 \end{bmatrix} \tag{22}$$

Computing the sum of each column of A_n according to Eq. (17) , we take the jth column as

$$\begin{aligned} & f_{1j} + f_{2j} + \cdots + f_{nj} \\ &= \frac{l - d_{1j}}{\sum_{t\in T_j} l - d_{tj}} + \frac{l - d_{2j}}{\sum_{t\in T_j} l - d_{tj}} + \cdots + \frac{l - d_{nj}}{\sum_{t\in T_j} l - d_{tj}} \\ &= \sum_{t\in T_j} \frac{l - d_{tj}}{\sum_{t\in T_j} l - d_{tj}} \\ &= 1 \end{aligned} \tag{23}$$

Therefore, Eq. (19) is convergent and satisfies

$$\sum_{m=1}^{n} SHR(N_m) = \sum_{m=1}^{n} R^{\alpha}(N_m) \tag{24}$$

where $\alpha \in [0, \eta]$.

6. Evaluation

In this section, we evaluate our architecture and conduct extensive experiments as follows:
(1) Using the feature extraction method with generalization ability, IDSTH can solve the out-of-sample problem (see Section 6.1).
(2) The efficiency of graph building using hash codes generated by IDSTH can be greatly improved with allowed accuracy loss (see Section 6.2).
(3) SHR can calculate the importance score for each node effectively and efficiently on large-scale datasets (see Section 6.3).
(4) SHR can help highlight and prepose those data whose semantic information account for higher proportion in original dataset (see Section 6.4).
(5) Our framework can deal with large-scale datasets and return a concise score (assessment result) based on the user's query, which assists the user to make a correct decision on subsequent operations with this dataset (see Section 6.5).

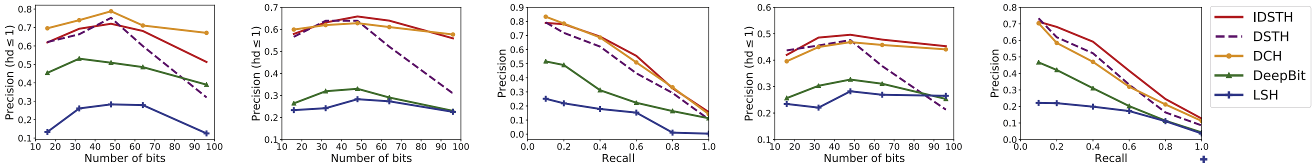
We implement the experiments (1) to (4) on the public datasets which include CIFAR-10 and STL-10. We summarize these datasets and corresponding preprocessing as follows:

- **CIFAR-10:** CIFAR-10 are labeled subsets of the 80 million tiny images dataset, which consists of 60,000 32 × 32 color images in 10 classes, with 6000 images per class. There are 5000 training images and 1000 test images in each class.
- **STL-10:** STL-10 is a subset of ImageNet dataset for image recognition, which consists of 10 classes, 5000 training images (500 images per class), 8000 test images (800 images per class) and 1,000,000 unlabeled images, each of size 96 × 96 pixels. We select 5000 training images and 8000 test images to complete our experiments.

Then we adopt large-scale real-world Tencent datasets to conduct the last experiments. Our evaluation is executed using Python tools including TensorFlow and Scikit-Learn library. Our experiments are run on two 10-core Intel Xeon E5-2640 machines with 128 GB of DDR4 memory. At last, we conduct the experiment on real-world dataset using 12 machines.

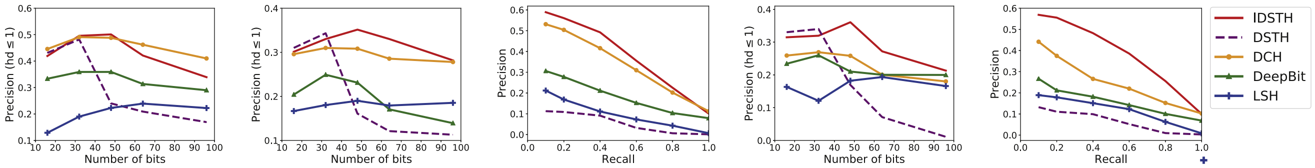
6.1. Generalization ability

In this section, we verify the generalization ability of IDSTH mapping hash by executing code length analysis (CLA) and precision–recall (PR) on CIFAR-10 and STL-10. We compare the state-of-the-art and the most classical semantic similarity hashing algorithms: DCH (the best supervised method), DeepBit (the best unsupervised method) and LSH (the classical data-aware method). In addition, we also compare DSTH algorithm.



(a) mAP CLA (b) mAP CLA without training cat (c) 48-bits PR without training cat (d) mAP CLA without training cat and automobile (e) 48-bits PR without training cat and automobile

Fig. 4. Code Length Analysis (CLA) and Precision–Recall (PR) curve on CIFAR-10.

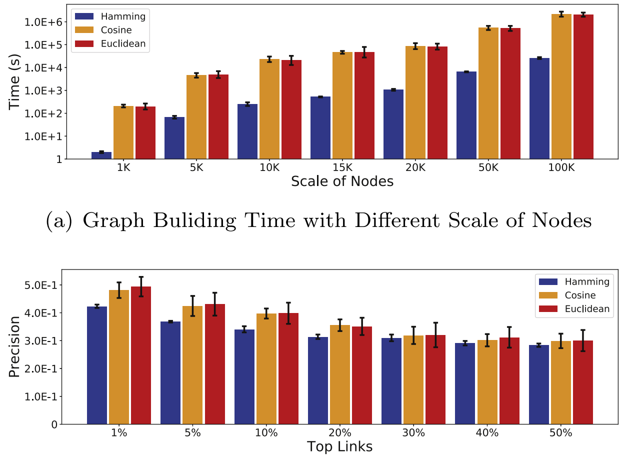


(a) mAP CLA (b) mAP CLA without training cat (c) 48-bits PR without training cat (d) mAP CLA without training cat and automobile (e) 48-bits PR without training cat and automobile

Fig. 5. Code Length Analysis (CLA) and Precision–Recall (PR) curve on STL-10.

To verify the generalization ability, we remove all cats and automobiles from the training dataset (On the testing dataset, if the algorithm can classify a cat as a dog and an automobile as a truck, these classification results are considered to be correct.) Fig. 4 shows the experimental results on CIFAR-10, where (a) denotes the average precision of the retrieval results of all categories when the *hd* $\leq$ 1 with different hash code length, (b) denotes the same code length analysis (CLA) without training cats, (c) denotes the precision–recall (PR) curve when (b) choosing 48-bits hash code, (d) denotes the CLA without training cats and automobiles, and (e) denotes the PR curve when (d) choosing 48-bits hash code. Fig. 5 shows the experimental results on STL-10, where (a)–(e) are similar to (a)–(e) of Fig. 4.

As shown in Fig. 4(a), in most cases, IDSTH is only inferior to DCH because supervised methods have advantage on classification. However, when the testing dataset contains one category data that has never been trained (shown in Fig. 4(b)), IDSTH shows superiority in most of the cases. Besides, as shown in Fig. 4(d), when the testing dataset contains two unacquainted categories data, IDSTH shows the overwhelming advantage compared with other methods. What is more, this phenomenon is more obviously reflected on STL-10 (because STL-10 owns a smaller training dataset, which weakens the advantage of supervised training). Therefore, IDSTH has an advantage in dealing with out-of-sample problems. In addition, we find IDSTH performs best at 48-bits hash code under various conditions, and Fig. 4(c)(e) also show the superiority on the PR curve at 48-bits hash codes. Therefore, we use 48-bits hash code in the follow-up experiments.



(a) Graph Buliding Time with Different Scale of Nodes

(b) Precision of Graph using Different Top Links with 100,000 Nodes

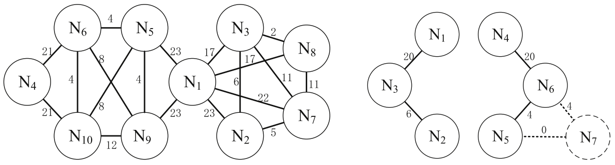
Fig. 6. Graph building time with different scale of nodes and precision of graph with 100,000 nodes using Hamming, Cosine and Euclidean distance.

6.2. Graph building efficiency

In this section, for verifying that building a graph by Hamming distance is more efficient than Cosine and Euclidean distance, we exhibit the time of graph building using three metrics with 48-bit vectors (48-bit hash codes and 48 float numbers). We randomly sample data from the overall data to form different data scales. Under each scale, we construct the graph for 20 times, and count the construction time cost of and precision of the graph. In order to ensure the fairness, we set $\Omega = 48$, making all nodes fully connected. As shown in Fig. 6(a), the horizontal coordinate represents the number of nodes while the ordinate represents the graph building time. With the same scale of nodes, the graph building time of Hamming distance is nearly 100 times less than that of Cosine and Euclidean, which shows that Hamming distance has overwhelming predominance over other two metrics in building a graph.

In order to compare precision of graph in three metrics, we choose more accurate links from top 1% to top 50% according to the weight of edges with 100,000 nodes. For example, we choose those edges on which the Hamming distance is smaller, while selecting the edges whose Cosine and Euclidean distance is larger. As shown in Fig. 6(b), Hamming distance is 0.070 lower than Euclidean at top 1% links in the worst case and 0.010 lower than Cosine at top 30% links in the best case in term of precision of graph. Averagely, Hamming distance is 0.038 lower than other two metrics in seven cases.

On the whole, there is not a marked difference of precision between three metrics, although hashing will bring certain loss to precision. However, Hamming distance has overwhelming predominance in building a graph in term of time cost. We



(a) Graph G_1 (b) Graph G_2

Fig. 7. Example graphs.

use hashing and Hamming distance in the follow-up work with comprehensive consideration of trade-off between efficiency and precision, since an acceptable margin of error is allowed.

6.3. SHR calculation

In this section, we verify that SHR can obtain reasonable importance score for each node on single and double connected domains respectively. Besides, we present the acceptable actual calculation cost of SHR under different number of nodes and iterations, indicating that SHR is able to adapt to large-scale scenes. In practice, we conduct experiments with 48-bit hash codes.

We prove feasibility using graph G_1 shown in Fig. 7(a). The results calculated by SHR are shown in Table 1 with $\eta = 65$, when we set $\varepsilon = 1.0\text{E}{-}10$ and $R^0(N_m) = 1$ where $m \in [1, 10]$. As shown in Fig. 7(a), N_1 has the most connections, while N_7 owns more edges where the Hamming distance is smaller relatively. Therefore, the results are deemed reasonable.

We prove reliability using graph G_2 shown in Fig. 7(b). Different from G_1, G_2 consists of 2 connected domains. The results calculated by SHR are shown in Table 2 with $\eta = 141$. Similarly, we set $\varepsilon = 1.0\text{E}{-}10$ and $R^0(N_m) = 1$ where $m \in [1, 6]$. Obviously, N_3 and N_6 get the same rank and play the most important role in their own connected domain. Furthermore, when we add N_7 which has the same hash code as N_5, the results will change shown in Table 2. It is easy to find that the ranks in the left domain do not change but the sum of ranks in the right domain has added one unit. Consequently, SHR is able to calculate the rank of each node in its own connected domain, without being influenced by other connected domains. Also, N_5 and N_7 own the same score, illustrating that those nodes which own the same hash code will get the same score.

For illustrating the performance of SHR, the number of nodes, the number of links, the time cost including calculating A in Eq. (22) and iterating, and the number of iterations are displayed with $\varepsilon = 1.0\text{E}{-}15$, $1.0\text{E}{-}11$, $1.0\text{E}{-}7$ and $\Omega = 24$ after graph building. As shown in Table 3, as the number of nodes increases, the number of iterations is relatively stable, since it is not determined by the scale of nodes and the main factor that causes the time cost of computing is the acquisition of A. In addition, the growth of time cost and number of links are acceptable with scale of nodes increasing. Even though the number of links exceeds 100 million, the number of iterations is very close to that of PageRank [8] proposed by Google, which shows that SHR algorithm is sufficient to deal with large-scale computing.

6.4. Predominant semantics

In this section, we verify SHR can highlight and prepose those data whose semantic information account for higher proportion in this section, which shows our algorithm has practical significance for assessment tasks. In the next experiment, if the ranked results are correct, those images whose semantic distribution account for higher proportion in original dataset will obtain larger scores and higher ranks. Thus, based on CIFAR-10 test set, under the premise that the amount of images of other classes remains unchanged, we choose one class as a study object to be added to the sample, making the amount of this class reach 20%, 30%, 40%, 50%, 60% and 70% respectively on the whole dataset. We collect proportion of this class in the top 5% and top 20% of ranked results in six cases mentioned above. We set $\varepsilon = 1.0\text{E}{-}7$ and choose $\Omega = 24$, 16 and 12 to conduct the experiments.

Fig. 8(a)(b) show the percentage with different Ω in the top 5% and top 20% of ranked results respectively when choosing cat as the study object. As shown in Fig. 8(a)(b), in all cases, SHR magnifies original proportion of cat (the part that goes beyond the blue column), indicating the efficiency of this algorithm. Detailedly, compared with $\Omega = 16$ or 24, setting $\Omega = 12$ yields better performance on the magnification of the cat percentage in the top 5% of ranked results, where the cat percentage is averagely 27.7% higher than original proportion in six cases. Among them, the best result exceeds the original proportion by 33.3% in the case

Table 1
Score and rank in graph G_1.

Node	Hash code	Score	Rank
N_1	FFFFFFFFFFFF	1.269	1
N_2	FFFFFF800000	1.078	3
N_3	FFFFFFFE0000	0.978	8
N_4	000000000000	0.993	6
N_5	C000007FFFFF	0.555	10
N_6	0000001FFFFF	1.048	4
N_7	FBFF7F8000E0	1.100	2
N_8	FFFFFF7E0080	0.960	9
N_9	C0003079FFFF	0.988	7
N_{10}	0300001FFE7F	1.026	5

Table 2
Score and rank in graph G_2.

Node	Hash code	$N_1 \sim N_6$		$N_1 \sim N_7$	
		Score	Rank	Score	Rank
N_1	1FFFFFFFFFFF	0.717	5	0.717	6
N_2	FFFFFF800000	0.822	4	0.822	5
N_3	FFFFFFFE0000	1.459	1	1.459	1
N_4	000000000001	0.711	6	0.506	7
N_5	C000007FFFFF	0.829	3	1.028	3
N_6	0000001FFFFF	1.459	1	1.436	2
N_7	C000007FFFFF	–	–	1.028	3

Table 3
Statistic of the number of nodes, the number of links, the time cost and the number of iterations with different ε when SHR is running.

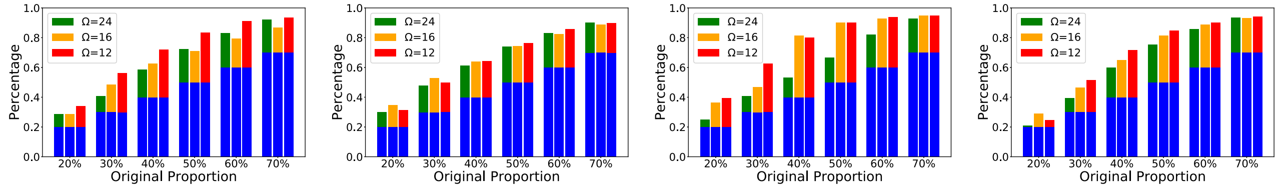
Node	Link	Time cost	Number of iterations ($\varepsilon =$)		
	$\Omega = 24$	Unit:s	1.0E−15	1.0E−11	1.0E−7
1k	305k	4 ± 0.08991	59	41	24
5k	7.5M	88 ± 0.2307	57	40	24
10k	29M	381 ± 0.8522	57	40	23
15k	66M	792 ± 1.333	57	40	24
20k	111M	1430 ± 2.028	58	40	24
50k	737M	8189 ± 5.465	55	38	22
100k	2.87G	29,484 ± 8.168	52	36	21

Table 4
Statistical results of classification by deep learning model on 1,000,000 images of Tencent dataset.

Class	Statistic	Proportion
People	661,254	66.13%
Animal	119,772	11.98%
Plant	24,330	2.433%
Building	100,012	10.00%
Automobile	19,673	1.967%
Pool	36,444	3.644%
Mountain	38,515	3.851%

Table 5
Effect comparison between our framework and deep model.

	Task	*hd*					DeepModel
		4	8	12	16	24	
Recommendation	A	1522	6952	29 368	41 996	16 668	661 254
	B	4167	10 731	46 731	74 618	391 742	724 099
	C	2150	6761	10 982	21 555	99 745	19 673
SSD adoption	A	1391	5332	17 747	28 139	90 121	21 116
	B	2710	5418	22 168	31 612	86 746	16 123
	C	61	102	187	248	495	188
Ratio	A	0.914	0.767	0.604	0.67	0.541	0.032
	B	0.65	0.505	0.474	0.424	0.221	0.02
	C	0.028	0.015	0.017	0.012	0.005	0.01
Subsequent mining time (s)	A	297.13	1458.021	6090.028	8784.37	34 781.16	13 711.032
	B	1205.021	3106.50	13 617.44	21 782.42	104 201.010	200 052.20
	C	368.52	1138.011	1801.47	3548.0051	16 634.33	3289.61



(a) Percentage of Cat in Top 5%

(b) Percentage of Cat in Top 20%

(c) Percentage of Automobile in Top 5%

(d) Percentage of Automobile in Top 20%

Fig. 8. Trend for percentage of cat and automobile in ranked result using different Ω with 48-bit codes. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

of 50%. In the top 20% of ranked results, choosing $\Omega = 12$ yields better performance in most of the cases, where the cat percentage is averagely 21.2% higher than original proportion in six cases. Among them, the best result exceeds the original proportion by 26.3% in the case of 50%.

Similarly, Fig. 8(c)(d) show the results choosing automobile as the study object. As is shown in Fig. 8(c)(d), SHR achieves the same effect. Detailedly, in the top 5% of ranked results, compared with other setting of Ω, the percentage of automobile shows the superiority in most of the cases while choosing $\Omega = 12$, which is averagely 31.7% higher than original proportion in six cases. Among them, the best result exceeds the original proportion by 40% in the case of 50%. Besides, in the top 20% of ranked results, the percentage of automobile also shows great superiority with $\Omega = 12$, which is averagely 31.0% higher than original proportion in six cases. Among them, the best result exceeds the original proportion by 34.7% in the case of 50%.

It should be explained that better precision and shorter time cost can be captured theoretically when Ω <12. However, the reduction of links causes too many isolated nodes all of whom get the same score, which may bring certain loss to the ranked results. Usually, with a larger scale of nodes, hash codes are more widely distributed, thus setting a smaller Ω will not result in too many isolated nodes. In our follow-up research, we intend to study this issue in depth.

As above experimental results show, both in the top 5% and 20% of ranked results, SHR effectively highlights and preposes the data whose semantic information account for higher proportion in original dataset after ranking. Thus, our SHR is correct and effective in practical applications.

6.5. Assessment of query

In this section, we verify that our framework can efficiently complete on-line assessment work according to the user's query on large-scale real image dataset. We apply our framework to real-world Tencent data source of a famous social platform. This data is a subset of QQ album data from 2017 to 2018 of Tencent. The size of data is around 5TB consisting of 1,000,000 images. Specially, according to the results shown in Section 6.4 that a smaller Ω is proved to be feasible at a million scale, we select $\Omega = 2$ to conduct this experiment.

We assess relevance of the dataset for three queries which include human intimacy as task-*A*, lovers traveling in the outskirts as task-*B*, and driving on road as task-*C* respectively. The query images are collected from Google search and their respective weights are given below. Fig. 9 displays the assessing process and results for above tasks. As shown in Fig. 9, the intimacy image representing the first task (query) matches three images whose ranks are high, so it is worth carrying out data mining on this dataset for the task-*A*. For task-*B* which matches two images contained in the dataset, the images of lovers have high scores and images about landscape own medium ranks. However, the weighted score of this task is relatively high, which shows this dataset can help analyze images about lovers traveling in the outskirts. Although there are three images that match the task-*C*, neither automobiles nor highways obtain high scores, so this dataset are not suitable for the task-*C*.

For exposing semantic information of the data source, we display the statistical results by previous classification model in Table 4. According to the classification results, except that the amount of images including people is obviously dominant, no more information can be captured for more specific assessment. Because of the limited expressive ability, classification cannot provide value judgment for diverse queries.

To further verify the correctness of our assessment, we show the efficiency of the mining algorithm according to our framework. Two sets of results returned by our framework with diverse *hd* and the deep model for above tasks are shown in Table 5. The statistical results include the number of recommended images that above two methods return, the number of suitable images that SSD [14] algorithm picks out from the recommended images, the proportion of suitable images in recommended images, and the time cost that SSD spends on the recommended images.

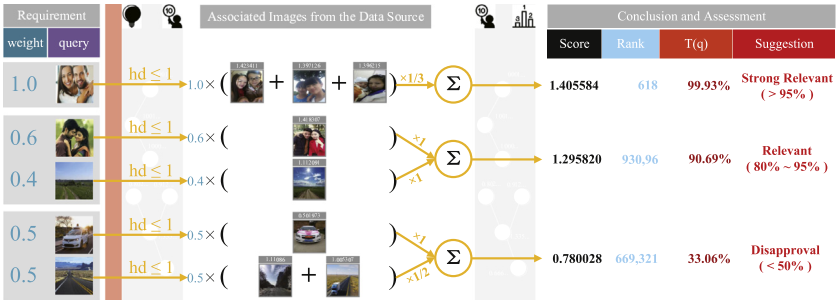


Fig. 9. The process of assessment on real-world dataset for three different queries.

Table 6
Time cost (unit:s) for assessment on Tencent dataset.

	Number of query images in a group (number of groups)				
	1 (1000)	2 (500)	4 (250)	5 (200)	8 (125)
$hd \leq 1$	1.21 ± 0.081	1.68 ± 0.17	3.75 ± 0.23	4.23 ± 0.26	7.37 ± 0.35
$hd \leq 2$	1.62 ± 0.047	2.72 ± 0.091	5.77 ± 0.16	6.62 ± 0.22	8.22 ± 0.35
$hd \leq 4$	2.33 ± 0.062	3.09 ± 0.12	6.18 ± 0.19	6.94 ± 0.24	9.38 ± 0.37

It is easy to find that, as *hd* becomes larger, the number of recommend images increases while the number of considered valid images is decreasing, because a larger *hd* will lead to a greatly ascending number of those weakly correlated images. Even so, our framework has an overwhelming predominance over deep model in terms of precision of the recommendation (ratio). This is because the semantic information which contains the features with generalization ability extracted by DSTH and the association analysis produced by SHR cannot directly captured by deep model.

Furthermore, we also find that although task-*B* has a relatively similar ranking query score with task-*A*, the number of its recommendation is greatly larger, because it is associated with two different semantics. At the same time, task-*C* does not get a large number of recommendation though also associated with two semantics, because the associated semantics account for a low proportion. Even so, for dominant data, our algorithm does not give more recommendation data than deep model, which reduces the analysis time cost for subsequent data mining. Our result benefits from the combined effects of features with generalization ability extracted by DSTH and association analysis produced by SHR, which reduces the number of rough recommended images about single object that deep model tends to return.

According to the ratio, even though setting $hd = 24$, the valid recommendation of task-*A*, task-*B* and task-*C* account for more than 50%, 22% and 0.5% respectively. This is completely corresponding to our importance rank and suggestion, indicating that our framework can truly and effectively expose the amount and entity of relevant data in the dataset.

Besides, we select 1000 different images from the Tencent data as query images to manifest the efficiency of assessment with different *hd*. In addition to counting the assessment efficiency by querying a single image, we also respectively randomly combine 2, 4, 5, 8 images together to collect the evaluation time cost. The results are shown in Table 6.

We find that evaluating single image is the most efficient, and the error is relatively smaller. As the number of images in each group increases, both the time cost and error are becoming larger. In addition, errors are relatively large when $hd \leq 1$, which is caused by the fact that some query images return none. It should be noted that we did not show the time cost according to different image sizes. In practice, we used cubic interpolation to resize all images into 256*256 to adapt to the input specification of ResNet. The time cost of resizing is negligible compared with that of evaluation.

7. Related works

Data quality assessment. Research on data quality (DQ) started in the 1990s and the initial research comes from group of MIT University led by Professor Richard Y. Wang. They defined DQ as "fitness for use" and used a two-stage survey to identify four categories containing fifteen DQ dimensions [15]. [16] summarized the most common dimensions and presented the approach to managing the implementation of quality related algorithms of a crawling search engine. [2] analyzed the challenges and importance of assuring the quality of big data. [1] as the latest research provided a DQ service for applications aiming at analyzing Big Data sources. They focused on the DQ Adaptor module and the reliability of DQ based on context-aware methodology.

Graph-based ranking. Calculating the importance score of each node is a special quantization method without clustering. It is more effective to get evaluation standards by ranking for each node globally. PageRank [8] considers out-degree of related nodes as impact factor for data ranking. [17] applies random walking to ranking community images for searching, which has achieved good results. [18] introduced the concept of probability to improve the RegEx in PageRank. However, above graph-based ranking algorithms focus on in-degree and out-degree, neglecting the weight on edges, resulting that they are not competent for quantization with Hamming distance. TextRank [19] and SentenceRank [7] take the weights on edges into consideration, both of which mentioned applying PageRank to improve their algorithms, but none of them give solution to image data analyzing and proof of convergence.

Graph embedding. Most graph embedding methods aim to learn representations for graphs. Some earlier works like LLE [20] and IsoMAP [21] utilized the eigenvectors as the network representation based on extracted feature vectors. More recent researches [22] designed two loss functions to capture the local and global network structure respectively. Furthermore, [23] extended this work to learn high-order information. Others such as DeepWalk [24], node2vec [25] used random walk and existing skip-gram to learn representations. However, all of them adopted shallow learning and failed to effectively capture the highly non-linear structure of networks. SDNE [26] designed a clear objective function to learn nodes embedding by preserving both the first-order and second-order proximity using deep neural network.

8. Conclusions

In this paper, we present a semantic-aware data quality assessment architecture, which can help the user perceive the semantic relevance between the requirements and image big data source in advance. Our IDSTH can effectively solve the out-of-sample problem to ensure the correctness of semantic extraction. Our designed SHR considers both the quantity of relevant data and the degree of semantic similarity comprehensively, which can give the assessment scores from a global perspective. Besides, combining deep learning and retrieval technologies, our developed architecture consists of off-line data analysis and on-line assessment, which ensures the real-time efficiency of assessment. Extensive experiments verify the accuracy and efficiency of our algorithm.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is supported by the Innovation Group Project of the National Natural Science Foundation of China No. 61821003 and the National Key Research and Development Program of China under grant No. 2016YFB0800402 and the National Natural Science Foundation of China No. 61672254.

References

[1] Danilo Ardagna, Cinzia Cappiello, Walter Samá, Monica Vitali, Context-aware data quality assessment for big data, Future Gener. Comput. Syst. 89 (2018) 548–562.

[2] Li Cai, Yangyong Zhu, The challenges of data quality and data quality assessment in the big data era, Data Sci. J. 14 (2015).

[3] Dong, Xin Luna, Divesh Srivastava, Big data integration, in: Proceedings of the 29th IEEE International Conference on Data Engineering (ICDE), 2013, pp. 1245–1248.

[4] H. Liu, Ming Shao, Sheng Li, et al., Infinite ensemble for image clustering, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016, pp. 1745–1754.

[5] Rashid Mehmood, Guangzhi Zhang, Rongfang Bie, Hassan Dawood, Haseeb Ahmad, Clustering by fast search and find of density peaks via heat diffusion, Neurocomputing 208 (2016) 210–217.

[6] Jingkuan Song, Lianli Gao, Li Liu, Xiaofeng Zhu, Nicu Sebe, Quantization-based hashing: a general framework for scalable image and video retrieval, Pattern Recognit. 75 (2018) 175–187.

[7] Ge, Shuzhi Sam, Zhengchen Zhang, Hongsheng He, Weighted graph model based sentence clustering and ranking for document summarization, in: Proceedings of the International Conference on Interaction Sciences, 2011, pp. 90–95.

[8] Lawrence Page, Sergey Brin, Rajeev Motwani, Terry Winograd, The PageRank Citation Ranking: Bringing Order To the Web, Stanford InfoLab, 1999.

[9] Ke Zhou, Yu Liu, Jingkuan Song, Linyu Yan, Fuhao Zou, Fumin Shen, Deep self-taught hashing for image retrieval, in: Proceedings of the 23rd ACM International Conference on Multimedia (MM), 2015, pp. 1215–1218.

[10] Yu Liu, Jingkuan Song, Ke Zhou, Lingyu Yan, Li Liu, Fuhao Zou, Ling Shao, Deep self-taught hashing for image retrieval, IEEE Trans. Cybern. 49 (2019) 2229–2241.

[11] Yue Cao, Mingsheng Long, Bin Liu, Jianmin Wang, Deep Cauchy hashing for hamming space retrieval, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 1229–1237.

[12] Hanjiang Lai, Yan Pan, Ye Liu, Shuicheng Yan, Simultaneous feature learning and hash coding with deep neural networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, pp. 3270–3278.

[13] Sergey Ioffe, Christian Szegedy, Batch normalization: Accelerating deep network training by reducing internal covariate shift, in: Proceedings of the International Conference on Machine Learning (ICML), 2015, pp. 448–456.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, C. Berg, SSD: Single shot multibox detector, in: Proceedings of the European Conference on Computer Vision (ECCV), 2016, pp. 21–37.

[15] Richard Y. Wang, Diane M. Strong, Beyond accuracy: What data quality means to data consumers, J. Manag. Inf. Syst. 12 (1996) 5–33.

[16] Shirlee-ann Knight, Janice Burn, Developing a framework for assessing information quality on the World Wide Web, Inf. Sci. 8 (2005) 159–172.

[17] Fabian Richter, Stefan Romberg, Eva Hörster, Rainer Lienhart, Multimodal ranking for image search on community databases, in: Proceedings of the ACM International Conference on Multimedia Information Retrieval, 2010, pp. 63–72.

[18] Yu Lei, Wenjie Li, Ziyu Lu, Miao Zhao, Alternating pointwise-pairwise learning for personalized item ranking, in: Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM), 2017, pp. 2155–2158.

[19] Mihalcea, Rada, Graph-based ranking algorithms for sentence extraction, applied to text summarization, Unt Sch. Works 20 (2004) 170–173.

[20] Sam T. Roweis, Lawrence K. Saul, Nonlinear dimensionality reduction by locally linear embedding, Science 290 (2000) 2323–2326.

[21] Joshua B. Tenenbaum, Vin De Silva, John C. Langford, A global geometric framework for nonlinear dimensionality reduction, Science 290 (2000) 2319–2323.

[22] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, Qiaozhu Mei, Line: Large-scale information network embedding, in: Proceedings of the 24th International Conference on World Wide Web (Www), 2015, pp. 1067–1077.

[23] Shaosheng Cao, Wei Lu, Qiongkai Xu, Grarep: Learning graph representations with global structural information, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management (CIKM), 2015, pp. 891–900.

[24] Bryan Perozzi, Rami Al-Rfou, Steven Skiena, Deepwalk: Online learning of social representations, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2014, pp. 701–710.

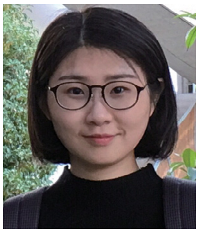
[25] Aditya Grover, Jure Leskovec, Node2vec: Scalable feature learning for networks, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016, pp. 855–864.

[26] Daixin Wang, Peng Cui, Wenwu Zhu, Structural deep network embedding, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD), 2016, pp. 1225–1234.

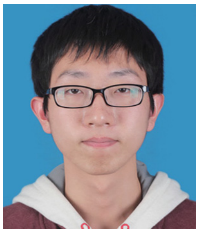
Yu Liu received the B.S. degree and M.S. degree from School of Computer Science and Technology, Wuhan Institute of Technology, Wuhan, China, in 2008 and 2012 respectively, and the Ph.D. degree in computer science from Huazhong University of Science and Technology (HUST) in 2017. Currently, he is a postdoctoral researcher in the Dept. of Software Engineering, HUST, China. His current research interests include machine learning, large-scale multimedia search, big data and storage, etc. He has published some papers in international journals and conferences, including ACM MM, FGCS, SIGMOD, MTAP, IEEE Transactions on Cybernetics, etc.

Yangtao Wang received the B.S. degree from Faculty of Mathematics and Statistics, Hubei University, Wuhan, China, in 2015. He is currently a Ph.D. candidate in Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology (HUST). His current research interests include differential privacy, machine learning, graph database, graph embedding, etc. He has published papers in international conferences including SIGMOD, ADMA, etc.

Ke Zhou is currently a professor of Huazhong University of Science and Technology (HUST). He received the B.S., M.S., and Ph.D. degrees in computer science and technology from HUST in 1996, 1999, and 2003 respectively. His main research interests include network/cloud storage, data security and service, parallel I/O. He has published more than 50 papers in international journals and conferences, including MSST, ACM MM, INFOCOM, ICS, ICPP, ICCD, SIGMOD, etc. He is a senior member of CCF and a member of IEEE.



Yujuan Yang received the B.S. degree from School of Computer Science and Information Engineering, Hubei University, Wuhan, China, in 2016. Currently, she is a M.S. candidate majoring in computer system architecture at Huazhong University of Science and Technology (HUST), China. Her current research interests include big data and storage, machine learning, technical architecture of large websites, etc.



Yifei Liu is currently a 3rd-year M.S. student at Huazhong University of Science and Technology (HUST). He is broadly interested in distributed file/storage systems including object storage, cloud storage, and NoSQL. Prior to HUST, he received the B.S. degree in computer science and technology from Huazhong Agricultural University in 2016.