Future Generation Computer Systems 86 (2018) 362–371

Contents lists available at ScienceDirect

Future Generation Computer Systems

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

Deep sentiment hashing for text retrieval in social CIoT

Ke Zhou [a], Jiangfeng Zeng [a,*], Yu Liu [b], Fuhao Zou [b]

[a] Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, Wuhan, China
[b] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan, China

HIGHLIGHTS

- We propose a supervised deep sentiment hashing method where hash labels and original sentiment labels are used as joint supervision information to generate the hash functions.
- Experiment results on several real-world datasets demonstrate the effectiveness of our approach for scalable sentiment-based hashing.

ARTICLE INFO

Article history:
Received 2 February 2018
Received in revised form 16 March 2018
Accepted 25 March 2018
Available online 25 April 2018

Keywords:
Text retrieval
Sentiment hashing
Social CIoT
Deep learning
Self-taught

ABSTRACT

Sentiment-based text retrieval is an urgent and valuable task due to the explosive growth of sentiment-expressed reviews from social networks like Twitter, Facebook, Instagram, etc. Social networks within the domain of Cognitive Internet of Things (CIoT) make it much easier to dynamically discover desirable services and valuable information. Information retrieval in social media is a daunting task which requires a lot of technical insights. As a powerful tool for large-scale information retrieval, hashing techniques have also been extensively employed for text retrieval. However, most existing text hashing methods are impractical for sentiment-expressed text retrieval mainly for three reasons: (1) the text representations are captured by shallow machine learning algorithms; (2) sentiment is rarely considered when measuring the similarity of two documents; and (3) unsupervised learning of hash functions is employed due to the lack of hash labels. To address these problems, in this paper, we put forward a general deep sentiment hashing model, which is composed of three steps. First, a hierarchical attention-based Long Short-Term Memory network (LSTM) is trained to obtain sentiment-specific document representations. Second, given the document embeddings, k-Nearest Neighbor (kNN) algorithm is used to construct a Laplacian matrix which is projected into hash labels via Laplacian Eigenmaps (LapEig) later. Third, we build a deep model for hash functions learning, which is supervised by both the generated hash labels and the original sentiment labels. Such joint supervision ensures that the ultimate hash codes produced by the learned hash functions maintain sentiment-level similarity. Experimental results turn out that the proposed approach achieves an effective and outstanding retrieval performance.

1. Introduction

The Internet of Things (IoT) connects all the things in the physical world to the Internet, and no one can deny the convenience and advantages that IoT brings us [1]. Along the development of IoT big data has successfully breeds a variety of techniques like cloud computing, deep learning, cognitive computing, etc. Since IoT systems depending only on IoT techniques suffer from a lot of issues, the integration of other complementary technologies and IoT is becoming an irresistible trend. CloudIoT enhances IoT with unlimited storage and powerful processing capacity by integrating cloud computing and IoT [2]. In order to better learn, think and understand both physical and social worlds, IoT should be empowered with cognitive capability and high-level intelligence with the explosive growth of cognitive science and artificial intelligence, which is called Cognitive Internet of Things (CIoT) [3]. It is known to all that the iron man's Javis is such a major breakthrough produced by CIoT. Although the existing derivations of CIoT like *APPLE*'s Siri cannot be so intelligent and groundbreaking, social networks within the CIoT domain make it much easier to dynamically discover desirable services and valuable information. Information retrieval in social media is a task of great importance.

Social CIoT aims to explore the manner in which CIoT techniques are exploited in social networks. Before going deep into the enabling techniques, let us first share one interesting CIoT

* Corresponding author.
E-mail addresses: k.zhou@hust.edu.cn (K. Zhou), jfzeng@hust.edu.cn (J. Zeng), lightyear416@163.com (Y. Liu), fuhao_zou@hust.edu.cn (F. Zou).

https://doi.org/10.1016/j.future.2018.03.047
0167-739X/

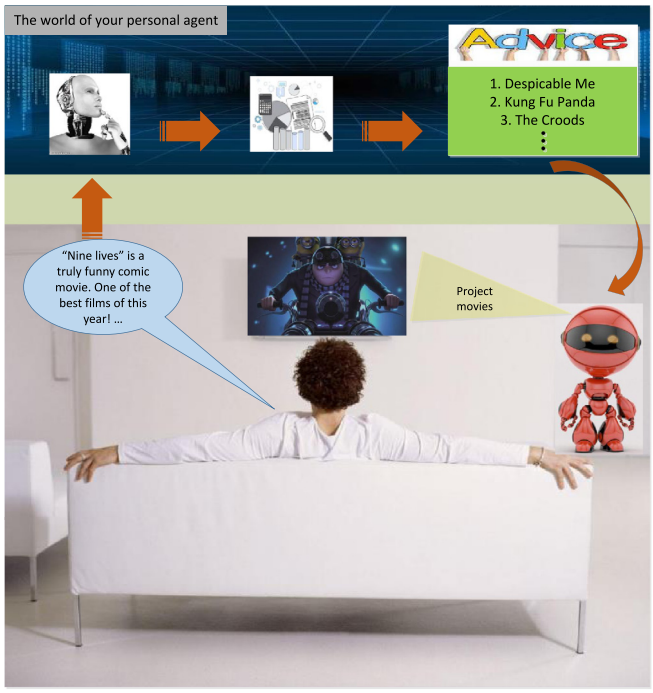

Fig. 1. An application scenario example in CIoT.

application scenario that will probably come into our daily life in the future:

Application scenario: Let us imagine that we live in a world where everyone has a personal agent system like iron man's Javis produced by CIoT. On a rainy Saturday, after five day's hard work, you would like to watch some funny comic movies for relaxation. But you have not the candidate movie list, and in fact it should not bother you to select several worth watching comic movies from thousands of movies. You just need to tell your personal agent system a movie review on one of your favorable comic movies once watched. For example, if your personal agent system is told that the "Nine Lives" is one of your favorable comic movies and the review about it is "One of the best films of the year! It is a truly funny comic movie. I laughed throughout the whole course. I like it very much.". Then the agent will try to sense and understand what sentiment you express in the review and response to you a list of movies commented with sentiment similar to the sentiment expressed in your review about the "Nine Lives". Fig. 1 vividly presents this example for illustration. There is nothing more convenient than just saying some words to your personal agent system. What a wonderful world we live!

Large-scale text retrieval has attracted more and more attention over the past decades. Due to an explosive growth in the amount of reviews from social networks, urgent requests are demanded for sentiment-based scalable text retrieval. It is well known that the star rating mechanism has been widely used in review websites. However, the star rating scales are not able to elaborately express the sentiment contained in reviews. For example, just given a review about a horror movie labeled with 5 stars (the biggest number), we do not know whether the review reveals a sentiment that it is a worth watching horror movie yet. We argue that modeling sentiment factor in text retrieval renders a much more convenient and sentiment-directed personalized recommendation service to customers and brings significant gains in financial revenues for e-commerce companies. Let us take movie rating website as an example. The movie rating website contains millions or billions of user reviews about movies and all these reviews are labeled with 1–5 star scales. From the user's perspective, the website should be equipped with an interface, which takes one sentiment-specific movie review as input, and retrieves sentimentally similar movie reviews for ultimate movie recommendation. To this end, the classification model and the sentiment hashing model are two options. We assume that the query movie review is "This is a truly remarkable horror movie. Very surprising. A truly astonishing vision." and is labeled with 5 stars. If we choose the classification model to address this task, it will obtain a large number of reviews predicted to score 5 stars no matter the candidate reviews are about horror movies, comedy movies, action movies or romance movies. Notedly, the retrieval results are invaluable and not instructive. Otherwise, the sentiment hashing model will produce some ranked reviews by measuring the similarity between the sentiment hash codes of this input review and the sentiment hash codes of other reviews. The returned reviews are supposed to be instructive and valuable since the number of retrieval results can be controlled via hamming distance. In theory, those reviews expressing such a sentiment that the horror movie is highly worth watching should be retrieved. By comparison, the sentiment hashing method captures much more fine-grained sentiments than the classification method. In a word, modeling sentiment into hashing is critical to large-scale sentiment-based text retrieval.

Hashing is the most popular retrieval approach applied to large-scale similarity search systems, and has become a promising technique for fast Approximate Nearest Neighbor (ANN) search over massive databases. It represents real-world high-dimensional data with compact binary codes and is able to compute similarity fast through comparing hamming distance, therefore being largely appreciated for its efficiency in both storage and computation. Hashing techniques have many practical applications. For example, [4] use hashing for social image completion. [5] learn unified binary codes for cross-modal retrieval via latent semantic hashing. There are two general paradigms in existing hashing approaches: data-independent and data-dependent. Data-independent hashing methods [6–8] have always been blamed for performing poorly with short hash codes. The latter, also known as learning-based hashing, leverages machine learning algorithms to produce similarity-preserving hash codes for original data by learning suitable hash functions from training procedure. Compared with data-independent hashing methods, learning-based hashing methods have become a stereotype because they are capable of encoding complex data into more semantic feature representations. We can further roughly group data-dependent hashing methods into unsupervised and supervised hashing. Unsupervised hashing methods [9–14] are intended to unsupervisedly mine inherent features from data for hash functions generation. By contrast, supervised hashing methods [15–21] have been demonstrated to be much more stable and accurate because the hash functions learned with supervised information can promote the hashing quality.

As stated above, learning robust and effective feature representations is crucial to generating powerful hash functions. Substantial practical applications have validated that deep neural networks (DNNs) have an overwhelming advantage over shallow models in feature representing. Several deep hashing models [22–28] are proposed for scalable image retrieval in recent years and have led to significant progress in retrieval performance. Inspired researchers attempt to build similar deep hashing models for text retrieval [29,30]. Undoubtedly, the fusion of hashing and DNN is an irresistible trend. However, most existing text hashing methods stagnate for poor feature representations using shallow models, and suffer from lack of hash labels so that it is difficult to devise supervised deep models to generate hash functions. What is worse, sentiment-based text hashing methods have not yet been explored.

Fig. 2 displays our proposed text retrieval framework in social CIoT, which includes data cognitive sensing, data preprocessing,

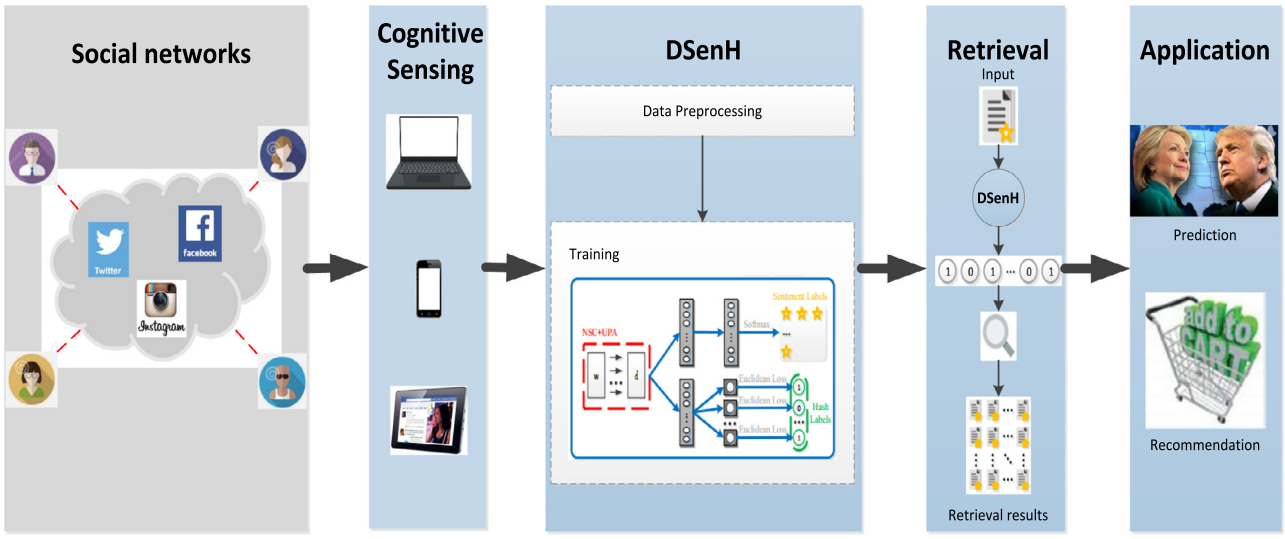

Fig. 2. Text retrieval framework in social CIoT.

model building and application. The core of the text retrieval framework is model building, and in this paper, we propose a novel deep sentiment hashing method for document retrieval, dubbed DSenH which can be completed in three steps. Firstly, we build a state-of-the-art attention-based LSTM model to generate sentiment-specific representations for documents. Secondly, we use kNN to construct a Laplacian matrix given the document embeddings and then convert the matrix into hash labels through Laplacian Eigenmaps (LapEig) [31]. This procedure is analogous to spectral hashing [10,32]. Finally, to learn hash functions supervisedly we establish a deep model which is structurally similar to the previous attention-based LSTM model but is trained independently. Detailedly, the hash functions learning is not only supervised by the hash labels generated from the second step, but also assisted by the original sentiment labels. Total loss is computed by proportionally adding the Euclidean loss from the generated hash labels and the cross-entropy loss from the original sentiment labels.

To summarize, our efforts are devoted to the following contributions.

- We propose a supervised deep sentiment hashing method that learns condensed binary codes for documents and achieves fast and outstanding document retrieval by sentiment. To the best of our knowledge, this is the first attempt that incorporates sentiment factor into semantic text hashing.
- We convert the document embeddings to hash labels by using kNN and LapEig and design a supervised deep model, where the hash labels and original sentiment labels are used as supervision information jointly to generate the hash functions.
- Experiments on several real-world datasets are conducted to demonstrate the effectiveness of our model for scalable sentiment-based text hashing.

The remainder of this paper is organized as follows. In Section 2, we review the two key components of deep sentiment hashing for document retrieval, i.e., hashing technologies and deep models for sentiment classification. Afterwards, we present the proposed deep sentiment hashing (DSenH) in Section 3. In Section 4, extensive experiments are conducted to demonstrate the superiority of the proposed algorithm. Finally, we draw a conclusion and envision the future in Section 5.

2. Related work

2.1. Hashing

Hashing, famous for bitwise XOR operation, usually can be classified into two categories: data-oblivious and data-aware. As a representative of the former, Locality Sensitive Hashing (LSH) [7,33] takes simple random projections as hash functions and consumes long binary codes to obtain good retrieval performance. Most recent researches concentrate attention on the latter which utilize a spectrum of machine learning algorithms to acquire similarity-preserving hash codes. Data-aware hashing methods are sensitive to intrinsic data distribution and can be divided into unsupervised and supervised hashing. In terms of unsupervised hashing, the Spectral Hashing (SpH) [10] constructs a spectral graph using kNN and executes min-cut on the graph through Laplacian Eigenmaps (LE). Graph hashing [13] utilizes the underlying manifold structure of data captured by graph representations. Self-Taught Hashing (STH) [12] defines a general two-stage framework: unsupervised learning of binary codes and supervised learning of hash functions, which has a great influence on later researches. [19] simplifies and optimizes the self-taught hashing algorithm based the work of [12]. In terms of supervised hashing, Kernel-Based Supervised Hashing (KSH) [15] uses pairwise relationship between samples as supervision information in the hash function learning stage and promotes the hashing quality. Binary Reconstructive Embedding (BRE) [16] minimizes the reconstructed error between the metric space and Hamming space. Asymmetric Discrete Graph Hashing (ADGH) [20] preserves the asymmetric discrete constraint and builds an asymmetric affinity matrix to learn compact binary codes. Note that the afore-mentioned hashing methods perform well but fail to fully preserve semantic similarity because the text representations are captured by shallow machine learning models.

Recently, deep hashing models have achieved promising results with powerful and semantic feature representations. Inspired by STH, Xia et al. proposed a supervised hashing method for image retrieval dubbed as CNNH [22]. They first compute approximate hash codes associated to a training image by decomposing the given pairwise similarity matrix. Then a CNN model is tailored to learn semantic feature representations for the input images as well as a set of hash functions. Xu et al. devised THC and three variants [29] exploring the power of text hashing via convolutional neural networks. THC combines word embeddings and position

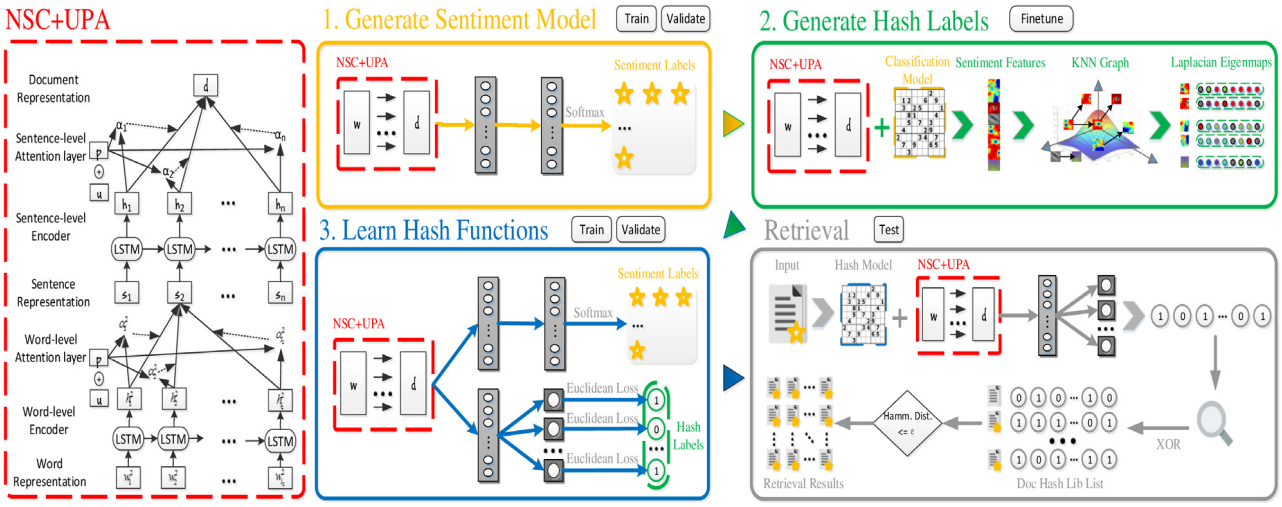

Fig. 3. Overview of the proposed deep semantic sentiment hashing approach for document retrieval. DSenH is achieved in three steps, i.e., training deep sentiment model, generating approximate hash labels and learning hash functions with joint supervision.

embeddings as features and binarizes them. Afterwards, word-level features are fed to a CNN model to learn hash functions. Chaidaroon et al. proposed VDSH [30] which combines variational autoencoder(VAE) and probabilistic generative models. VDSH uses VAE to produce binary codes unsupervisedly. Most existing deep text hashing models obey the two-step scheme defined by STH, but they suffer from poor-quality binary codes which are obtained either over pure word-level features or by unsupervised models.

2.2. Deep sentiment classification

Deep neural networks win a reputation in Computer Vision (CV) and Natural Language Processing (NLP). Kim adopted convolutional neural networks (CNN) to learn sentence representations and achieved significant success in sentence-level sentiment classification [34]. As dedicated to processing sequential data, recurrent neural networks (RNN) have shown great promise in many NLP tasks. Over the past years, most advanced deep NLP models have been established and can be summarized as a four-step formula: embed, encode, attend and predict. Tai et al. investigated tree-structured long-short term memory networks (TreeLSTM) [35], which represents sentence based on the syntactic dependency parsing tree. Xu et al. devised cached long short-term memory networks [36] for document-level sentiment classification, which introduces a cache mechanism to divide memory into several groups with different forgetting rates and thus enables the network to capture the overall sentiment information better within a recurrent unit. In view of the hierarchical structure of documents, some hierarchical models were built to tackle document-level sentiment analysis [37,38]. Attention mechanism has been proved to achieve superior performance in substantial applications, such as image generation [39,40], machine translation [41,42], image caption [43] and natural language inference [44]. Yang et al. introduced attention mechanism to select important word-level and sentence-level features hierarchically [45]. Recently, there emerged two models which took the global user preferences and product characteristics into consideration and were proved to be state-of-the-art [46,47].

3. The proposed approach

Drawing inspirations from [12,19], our proposed approach follows the two-stage paradigm and can be completed in three steps: training deep sentiment model, generating approximate hash labels and learning hash functions with joint supervision. The first two steps correspond to the unsupervised learning of binary codes and the last step is devoted to the supervised learning of hash functions. A high-level illustration of our proposed approach is shown in Fig. 3.

3.1. Step 1: training deep sentiment model

We first formulate the problem of review sentiment classification. Suppose a user $u \in R^{d_u}$ from U writes a review d about a product $p \in R^{d_p}$ from P. The review d_i is composed of n sentences $\{S_1, S_2, \ldots, S_n\}$. The jth sentence S_j contains l_j words. We take a set of training reviews $D = \{(d_1, y_1), (d_2, y_2), \ldots, (d_n, y_n)\}$ as inputs, where d_i is a document-level review and $y_i \in Y$ represents the sentiment class (e.g., $Y = \{1, 2, 3, 4, 5\}$ indicates the ratings from "one star" to "five stars"). The goal of document-level review sentiment classification is to predict the sentiment classes given reviews.

NSC+UPA [47] not only encodes words and sentences of one review hierarchically, but also introduces the global user preferences and product characteristics as attentions over different semantic levels of a document, therefore producing robust and semantic document embeddings. Let us take comments labeled with 1–5 star rating scales as an example. Given one review "its great", users of different tempers might mark 4 or 5 stars without a universally accepted evaluation criteria. And comments towards high-quality products (e.g. IPHONE) tend to score higher than those towards poor-quality products. Consequently, modeling the global user preferences and product characteristics contributes to building a smarter sentiment classifier than before.

As illustrated in Fig. 3, NSC+UPA runs in a hierarchical manner. Both word-level component and sentence-level component are processed with a sequence encoder followed by an attention layer.

3.1.1. Word-level encoder

For each sentence, we aim to generate a low-dimensional semantic sentence vector. We pre-train word embeddings by $word2vec$ [48] and map the tth word of the jth sentence S_t into its embedding $w_t^j \in R^{d_w}$. Then all the word embeddings $\{w_1^j, w_2^j, \ldots, w_{l_j}^j\}$ are fed to LSTM to generate a hidden output for each word, i.e, $\{h_1^j, h_2^j, \ldots, h_{l_j}^j\}$. Thus, we need to briefly provide background on Long Short-Term Memory networks (LSTMs) and illustrate the equations since LSTM is adopted as our sequence encoder.

Long Short-Term Memory networks (LSTMs) are explicitly designed to tackle the long-term dependencies since the standard recurrent neural networks (RNNs) suffer from gradient vanishing or exploding problem [49]. The key to LSTM is the cell state which can be regulated by three gates, i.e., input gate, forget gate and output gate. Formally, given the input word embedding w_t^j which vectorizes the tth word of the jth sentence, current cell state c_t and current hidden value h_t^j in a standard LSTM can be updated as follows:

$$i_t = \sigma(W_i^{(w)} \cdot [w_t^j, h_{t-1}^j] + b_i^{(w)}) \tag{1}$$

$$f_t = \sigma(W_f^{(w)} \cdot [w_t^j, h_{t-1}^j] + b_f^{(w)}) \tag{2}$$

$$o_t = \sigma(W_o^{(w)} \cdot [w_t^j, h_{t-1}^j] + b_o^{(w)}) \tag{3}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tanh(W_c^{(w)} \cdot [w_t^j, h_{t-1}^j] + b_c^{(w)}) \tag{4}$$

$$h_t^j = o_t \odot tanh(c_t) \tag{5}$$

where i, f and o are input gate, forget gate and output gate respectively, which selectively add new information and discard useless information in each iteration via the sigmoid function, i.e., σ in the equations. $\Theta_{word}^{lstm} = \{W_i^{(w)} \in R^{d\times(d+d_w)}, W_f^{(w)} \in R^{d\times(d+d_w)}, W_o^{(w)} \in R^{d\times(d+d_w)}, b_i^{(w)} \in R^d, b_f^{(w)} \in R^d, b_o^{(w)} \in R^d\}$ are denoted as the parameters of LSTM to be learned during training. $\odot$ stands for element-wise multiplication and $[w_t^j, h_{t-1}^j]$ means the concatenation operation.

3.1.2. Word-level attention

Evidently, not all words contribute equally to the sentence representation, so the global user preferences and product characteristics are brought in to select the crucial components over hidden values of each sentence. Formally, the enhanced sentence embedding is a weighted sum of the words' hidden states as:

$$s_j = \sum_{t=1}^{l_j} \alpha_t^j h_t^j \tag{6}$$

where α_t^j measures the importance of the tth word in terms of current user and product. We define the attention probability distribution α_t^j as:

$$\alpha_t^j = \frac{\exp(e(h_t^j, u, p))}{\sum_{k=1}^{l_j} \exp(e(h_k^j, u, p))} \tag{7}$$

$$e(h_t^j, u, p) = \eta^T tanh(W_h^{(w)} h_t^j + W_u^{(w)} u + W_p^{(w)} p + b^{(w)}) \tag{8}$$

where $\Theta_{word}^{att} = \{W_h^{(w)} \in R^{d\times d}, W_u^{(w)} \in R^{d\times d_u}, W_p^{(w)} \in R^{d\times d_p}, b^{(w)} \in R^d\}$ are model parameters. η^T denotes its transpose.

3.1.3. Sentence-level encoder

By analogy, all the sentence representations $\{s_1, s_2, \ldots, s_n\}$ of one document d are fed into LSTM to generate a hidden output for each sentence, i.e., $\{h_1, h_2, \ldots, h_n\}$. Similarly, $\Theta_{sent}^{lstm} = \{W_i^{(s)} \in R^{d\times(d+d)}, W_f^{(s)} \in R^{d\times(d+d)}, W_o^{(s)} \in R^{d\times(d+d)}, b_i^{(s)} \in R^d, b_f^{(s)} \in R^d, b_o^{(s)} \in R^d\}$ are denoted as the parameters of LSTM to be learned during training.

3.1.4. Sentence-level attention

Similarly, we utilize user and product attention to endow each sentence embedding with a weight of different importance. Denote the output of sentence-level LSTM as:

$$d = \sum_{i=1}^{n} \alpha_i h_i \tag{9}$$

where α_i measures the importance of the ith sentence in terms of current user and product and can be calculated similar to the word-level attention. Here, $\Theta_{sent}^{att} = \{W_h^{(s)} \in R^{d\times d}, W_u^{(s)} \in R^{d\times d_u}, W_p^{(s)} \in R^{d\times d_p}, b^{(s)} \in R^d\}$ are model parameters.

3.1.5. Sentiment classification

We regard the document representations as features for sentiment classifier. First of all, a non-linear layer is used to project d into the target space of C classes.

$$\widehat{d} = tanh(W_c d + b_c) \tag{10}$$

Afterwards, a softmax layer is utilized to obtain the document sentiment distribution.

$$p_c = \frac{\exp(\widehat{d}_c)}{\sum_{c=1}^{C} \exp(\widehat{d}_c)} \tag{11}$$

where p_c represents the probability that document d belongs to class c. In NSC+UPA, the cross-entropy loss is defined as the objective function $L_{SENTIMENT}$.

$$L_{SENTIMENT} = -\sum_{d\in X} \sum_{c=1}^{C} p_c^g(d) \cdot \log(p_c(d)) + \mu\Omega\left(\Theta^{(sentiment)}\right) \tag{12}$$

where, for training dataset X, p_c^g is the gold probability of class c with ground truth being 1 and others bedding 0, $\Theta^{(sentiment)} = \{\Theta_{word}^{lstm}, \Theta_{word}^{att}, \Theta_{sent}^{lstm}, \Theta_{sent}^{att}, W_c, b_c\}$ represents all parameters. We denote $\Omega(\cdot)$ as a regularization function to sum up the l_2-norm of each parameter in $\Theta^{(sentiment)}$. $\mu > 0$ is a trade-off parameter. The deep sentiment model can be successfully trained end-to-end via backpropagation (BP).

3.2. Step 2: generating approximate hash labels

Once NSC+UPA is trained, we utilize the model to predict document embeddings for training dataset, test dataset and validation dataset respectively. Then we transform the document embeddings of training dataset, test dataset and validation dataset into hash labels individually, retaining relative distances among data from high-dimension vector space to low-dimension hamming space. Mathematically, $\{d_i\}_{i=1}^N \in R^M$ denotes the document embeddings. d_i and y_i represents the ith sample and its hash label respectively. $y_i \in \{0, 1\}^l$ and l is the number of bits. We set y_i^ρ as the ρ-th element of y_i, and it equals 1 if the ρ-th bit is 1, or 0 otherwise. The hash code set for N samples can be represented as $[y_1, y_2, \ldots, y_N]^T$.

In this procedure, given the document embeddings, we construct a $N \times N$ local similarity matrix W via k-Nearest Neighbor (kNN). We formulate W as:

$$W_{ij} = \begin{cases} 0 & \text{if } N_k(d_i, d_j) \text{ is false,} \\ \dfrac{d_i^T d_j}{\|d_i\| \cdot \|d_j\|} & \text{otherwise.} \end{cases} \tag{13}$$

where $N_k(d_i, d_j)$ represents whether the ith sample and the jth sample are neighbors in the k-nearest set. Furthermore, we apply diagonal matrix

$$D_{ii} = \sum_{j=1}^{N} W_{ij} \tag{14}$$

Table 1
Dataset description.

Datasets	#classes	#docs	#users	#products	#docs/user	#docs/product	#sens/doc	#words/sen	#voc
IMDB	10	84,919	1310	1635	64.82	51.94	16.08	24.54	105,373
Yelp13	5	78,966	1631	1633	48.42	48.36	10.89	17.38	48,957
Yelp14	5	231,163	4818	4194	47.97	55.11	11.41	17.26	93,197

Meanwhile, we calculate Hamming distance between y_i and y_j as:

$$H_{ij} = \|y_i - y_j\|^2/4 \tag{15}$$

Similar to SpH, we define an object function ζ to minimize the weighted average Hamming distance.

$$\zeta = \sum_{i=1}^{N}\sum_{j=1}^{N} W_{ij}H_{ij} \tag{16}$$

To calculate ζ, we transform it to $\xi = tr(Y^T LY)/4$, where $L = D - W$ is the Laplacian matrix and $tr(\cdot)$ means the trace of matrix. At last, we transform ξ to LapEig problem ψ shown in Eq. (17) by slacking constraint $y_i \in \{0, 1\}^t$, and obtain the optimal t-dimensional real-valued vector $\tilde{y}$ to represent each sample.

$$\psi = arg\ \min_{\tilde{Y}} \quad Tr(\tilde{Y}^T L\tilde{Y}) \quad s.t. \begin{cases} \tilde{Y}^T D\tilde{Y} = I \\ \tilde{Y}^T D1 = 0 \end{cases} \tag{17}$$

where $Tr(\tilde{Y}^T L\tilde{Y})$ gives the real relaxation of the weighted average Hamming distance $Tr(T^T LY)$. The solution of this optimization problem is given by $\tilde{Y} = [v_1, \ldots, v_t]$ whose columns are the t eigenvectors corresponding to the smallest eigenvalues of following generalized eigenvalue problem. The solution of ψ can be transformed to

$$Lv = \lambda Dv \tag{18}$$

where vector v are the t eigenvectors corresponding to the t smallest eigenvalues (nonzero).

Then, we convert the t-dimensional real-valued vectors $\tilde{y_1}, \ldots, \tilde{y_n}$ into binary codes according to the threshold. We set ε^ρ to present the threshold and $\tilde{y_i^\rho}$ equivalent to the ρ-th element of $\tilde{y_i}$. The hash label as final result value of y_i^ρ is

$$y_i^\rho = \begin{cases} 1 & \tilde{y_i^\rho} \geqslant 0, \\ 0 & otherwise. \end{cases} \tag{19}$$

3.3. Step 3: learning hash functions with joint supervision

In this phase, we conduct a supervised end-to-end training to generate hash functions. The new model is structurally similar to the sentiment model but is trained independently. As described in Fig. 3, two mutually independent branches are devised after we obtain the document embeddings. The first branch is one non-linear MLP layer plus softmax layer for sentiment classification. The other branch is slice networks, which have been proved to be more effective than one MLP layer for learning hash function. Different from [50], we slice a M-dimensional document embedding to k groups and then map each group to q elements. The output of our slice networks is the hash vector with length $k \times q$. Due to the unsteady distribution of the output, directly using threshold value to binarize the output leads to a large bias. So we adopt an activation function to decrease the bias. Here we choose $tanh(\theta) = (e^\theta - e^\theta)/(e^\theta + e^\theta)$. Denote the activated output o as one $N \times l$ matrix (l equals $k \times q$), y is the corresponding hash labels obtained in step 2. We define the loss function as follows:

$$L_{SLICE}\ (o) = \sum_{i=1}^{N}\sum_{\rho=1}^{l} \left\| o_i^{(\rho)} - y_i^{(\rho)} \right\|_2^2 \tag{20}$$

To sum up, we establish a new model, which is not only supervised by the hash labels generated from second step, but also assisted by the original sentiment labels. We train the two branches simultaneously and set different parameters for the two losses, α and β, where $0 < \alpha, \beta < 1$ and $\alpha + \beta = 1$. As demonstrated in our experiments, we find that using the original sentiment labels as auxiliary supervision information enhances the quality of final hash codes. Here we set $\alpha = 0.8$ and $\beta = 0.2$ considering the predominant role of the Euclidean loss from the generated hash labels and the auxiliary role of the cross-entropy loss from the original sentiment labels. VDSH-SP [30] has validated the effectiveness of this manner. Thus the final loss is computed as follows:

$$L_{TOTAL} = \alpha \cdot L_{SLICE} + \beta \cdot L_{SENTIMENT} \tag{21}$$

We use back-propagation (BP) for learning and stochastic gradient descent (SGD) to minimize the total loss.

4. Experiments

4.1. Experimental settings

We evaluate the proposed approach on three real-world datasets, and the statistics of the datasets are summarized in Table 1. Yelp 2013 and Yelp 2014 are restaurant review datasets labeled with 1–5 star scales which are derived from Yelp Dataset Challenge of year 2013 and 2014 respectively. IMDB, a popular movie review dataset, consists of 84,919 movie reviews labeled with 1–10 star scales and the average sentence number of each review is larger than the number of two Yelp review datasets. All the datasets are publicly available and we pre-process the datasets in the same way as Chen et al. [47] did.

In our experiments, each dataset is roughly split into three parts in the proportion of 8:1:1 — a training dataset (80%), a held-out validation dataset (10%), and a test dataset (10%). It is worth mentioning that all the three datasets are unbalanced. We pre-train the 200-dimensional $word2vec$ embeddings on each dataset separately with SkipGram [48], and initialize the user embedding and product embedding with a 200-dimensional zero vector. We adopt an attention-based LSTM model to extract a 288-dimensional feature vector to represent each document. Our experiments are conducted with a batch size of 32 documents, l_2-regularization weight of 0.00001 and initial learning rate of 0.05 for AdaDelta.

4.2. Comparison experiments

We compare the performance of the proposed DSenH against seven state-of-the-art hashing methods, including data-unaware LSH [33], self-taught STH [12], three supervised methods using shallow machine learning algorithms (CCA-ITQ [14], KSH [15], BRE [16]), CNN-based text hashing THC-IV [29] and VDSH-SP combining VAE and probabilistic generative models [30].

Four evaluation metrics are utilized to evaluate the quality of hashing codes, i.e., mean Average Precision (mAP), Precision–Recall curves, Precision curves within Hamming distance 2, and Precision curves w.r.t. different number of top returned samples. To make a fair comparison, all of the methods conduct experiments

Table 2
mAP of the top 100 and 500 retrieved documents on three datasets with different number of bits. The bold font denotes the best results correspondingly.

Top-K	Methods		Yelp13			Yelp14			IMDB		
			8 bits	32 bits	48 bits	8 bits	32 bits	48 bits	8 bits	32 bits	48 bits
@100	CCA-ITQ		0.3591	0.3912	0.3873	0.4107	0.4299	0.4379	0.1762	0.1809	0.1777
	KSH		0.4008	0.4230	0.4331	0.4702	0.4881	0.4809	0.3033	0.3188	0.3294
	THC-IV		0.5516	0.5766	0.5892	0.5661	0.5758	0.5989	0.5197	0.5210	0.5211
	VDSH-SP		**0.5903**	0.6068	0.6092	0.6344	0.6575	0.6589	**0.5889**	**0.5913**	0.5822
	DSenH with	$\alpha = 0.2$	0.5522	0.5771	0.5833	0.6644	0.6667	0.6637	0.5066	0.5136	0.5117
		$\alpha = 0.5$	0.5794	0.6003	0.5991	0.6701	0.6712	0.6652	0.5302	0.5523	0.5382
		$\alpha = 0.8$	0.5881	**0.6218**	**0.6155**	**0.6711**	0.6690	0.6656	0.5741	0.5847	**0.5842**
		$\alpha = 1.0$	0.5800	0.6066	0.5916	0.6698	**0.6711**	**0.6693**	0.5520	0.5601	0.5577
@500	CCA-ITQ		0.2729	0.3225	0.3197	0.1986	0.1999	0.2039	0.0811	0.0921	0.0830
	KSH		0.2813	0.3333	0.3190	0.2907	0.3007	0.3019	0.1501	0.1509	0.1530
	THC-IV		0.4092	0.4312	0.4323	0.3816	0.3818	0.3907	0.3199	0.3233	0.3285
	VDSH-SP		0.4480	0.5253	0.5166	0.4907	0.5573	0.5716	**0.3519**	0.3398	0.3412
	DSenH with	$\alpha = 0.2$	0.4404	0.5100	0.5011	0.5269	0.5286	0.5266	0.3296	0.3320	0.3313
		$\alpha = 0.5$	**0.4833**	0.5315	0.5152	0.5377	0.5318	0.5328	0.3415	0.3438	0.3427
		$\alpha = 0.8$	0.4706	**0.5538**	**0.5205**	**0.5762**	**0.5829**	**0.5747**	0.3448	**0.3471**	**0.3455**
		$\alpha = 1.0$	0.4669	0.5521	0.5193	0.5755	0.5815	0.5643	0.3449	0.3458	0.3451

Table 3
Average F-measure score and precision for different parameters configuration of net.

Dataset	Net setting	Ave F_a		Precision
	Group:Map	$a = 0.5$	$a = 1$	H.D. ≤ 2
Yelp13	24:(12 → 1)	**0.62627**	**0.56548**	0.56901
	12:(24 → 2)	0.62405	0.56429	**0.57597**
	48:(6 → 1)	0.69680	0.60831	**0.96219**
	12:(24 → 4)	**0.70093**	**0.61068**	0.95682
Yelp14	24:(12 → 1)	0.63468	0.57063	0.58254
	12:(24 → 2)	**0.63546**	**0.57083**	**0.58433**
	48:(6 → 1)	0.69273	0.60582	0.94052
	12:(24 → 4)	**0.69931**	**0.60986**	**0.94290**
IMDB	24:(12 → 1)	**0.45930**	**0.44980**	0.39426
	12:(24 → 2)	0.45787	0.44862	**0.39460**
	48:(6 → 1)	0.52201	0.49352	0.48469
	12:(24 → 4)	**0.52271**	**0.49403**	**0.48998**

on identical training, validation and test datasets. We set the test dataset to be the test query set.

We conduct some experiments to verify the effect of our joint supervision and decide the proportion of each loss, i.e. the values of α and β. The mAP results are listed in Table 2, which show that the proposed approach substantially outperforms all the compared methods and is much more stable compared with other approaches. As representatives of deep hashing methods, THC-IV, VDSH-SP and our proposed DSenH score 15% higher on average than hashing techniques based on shallow machine learning algorithms. In comparison with THC-IV, our proposed DSenH with $\alpha = 0.8$ and $\beta = 0.2$ achieves absolute increases of 3.6%, 8.8%, and 6.0% in average mAP@100 and absolute increments of 9.0%, 19.3%, and 2.2% in average mAP@500 for different bits on Yelp13, Yelp14, and IMDB respectively. In comparison with VDSH-SP, DSenH with $\alpha = 0.8$ and $\beta = 0.2$ scores 0.6% and 1.5% higher in average mAP@100 and scores 1.8% and 3.8% higher in average mAP@500 for different bits on Yelp13 and Yelp14, respectively. With respect to the performance on IMDB, DSenH and VDSH-SP are definitely on a par. The main reason why the results in mAP@500 degrade by a large margin compared with those in mAP@100 is that our three datasets are unbalanced.

As shown in Fig. 4, DSenH exhibits better performance than the other counterparts on three datasets in most cases, especially with hash codes length of 48. Subfigure (a) and (d) indicate that our model is superior to others on Yelp13 and Yelp14 with hash codes length of 48, and subfigure (g) shows that our model prevails over others on IMDB when the length of hash codes is more than 12. It can be seen from subfigure (b), (e) and (h) that our *PR* curve degrades much slower than others on all datasets. Notably, subfigure (c) and (f) reveals that DSenH achieves higher accuracy with the same number of top returned samples on Yelp13 and Yelp14. From subfigure (i), we can see that DSenH performs much more stable and better than others in general though THC-IV surpasses DSenH partly.

In our experiments, a 288-dimensional document embedding is sliced to k groups and then map each group to q elements. By reference to DSTH [25] and [50], we investigate two kinds of mapping styles on each dataset when devising our slice networks: single mapping and multiple mapping. For example, if we want to produce hash codes with 24 bits, we either set k and q to 24 and 1 respectively or to 12 and 2 respectively. The traditional F-measure is the harmonic mean of the precision P and the recall R, and can be formulated as:

$$F_a = \frac{(a^2 + 1) * PR}{a^2 * P + R} \tag{22}$$

where $0 < a < 1$ means that the precision P contributes more than the recall R and $a = 1$ means that the precision P and the recall R contribute equally. As shown in Table 3, in terms of hash codes with 48 bits, F_1 scores 0.3% higher and $F_{0.5}$ scores 0.4% higher on average using multiple mapping style. As far as precision within hamming radius 2, multiple mapping style performs better than single mapping style is concerned and brings about 0.23% improvement on average.

In order to explicitly verify the retrieval performance of our proposed DSenH, we manually create a query dataset of 100 movie reviews based on the IMDB dataset, which contains 10 movie reviews per class and can be clearly told the sentiments. In detail, we select 10 movie reviews per class from the IMDB dataset, and remove redundancy descriptions about movie details. The main reasons why we need to manually create such a query dataset for testing are: (1) the IMDB dataset is heavily unbalanced; and (2) the movie reviews in IMDB dataset are long sequences which contains a lot of sentences describing the details of movies, thus making it difficult to acquire the sentiment at a quick glance. As displayed in Fig. 5, we take a review labeled with 10 stars as the query sample, and return top 9 movie reviews, where 6 reviews underlined in red are sentimentally equivalent to the query sample. Let us look back at the CIoT application scenario. Your personal agent system will recommend movies to you according to the sentimentally similar reviews produced by our proposed approach.

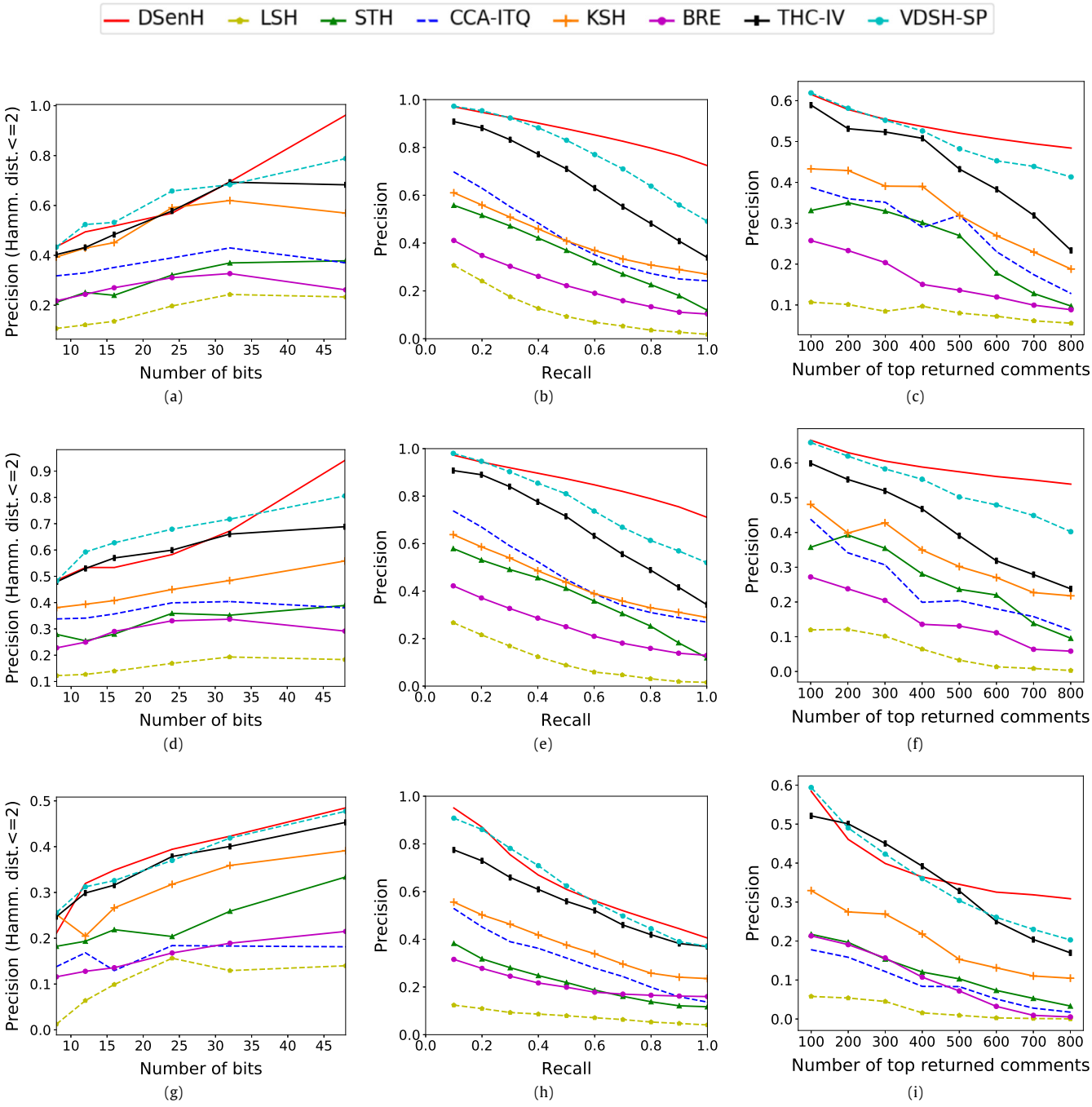

Fig. 4. The results on Yelp13, Yelp14 and IMDB. (a)–(c) show the results on Yelp13; (d)–(f) show the results on Yelp14; (g)–(i) show the results on IMDB; (a), (d) and (g) display precision curves within Hamming radius 2; (b), (e) and (h) display precision–recall curves of Hamming ranking with 48 bits; (c), (f) and (i) display precision curves with 48bits w.r.t. different number of top returned samples.

5. Conclusion and future work

With the rapid evolution of AI techniques, especially deep learning architectures, the Cognitive Internet of Things (CIoT) has shown a promising prospect. In this paper, we present a novel approach to deal with scalable sentiment-based text retrieval in social CIoT. As far as we know, it is the first attempt in the literature to address this task. The proposed DSenH can be achieved in three steps. The first two steps produce the approximate hash labels and prepare for the succeeding step which aims to learn hash functions for the final hash codes generation and is supervised by the hash labels and sentiment labels jointly. Finally, extensive experiments conducted on three well-known datasets have demonstrated that the ultimate hash codes stem from semantic sentiment-oriented representations and can preserve sentiment-level similarity well.

Although our proposed method can achieve compelling results when addressing scalable sentiment-based text retrieval in social CIoT, the hash codes produced via DSenH are sentiment-oriented, thus preserving less topic information than sentiment information. In other words, limited by the experimented datasets which are only labeled with sentiment tags and lack topic tags, the DSenH fails to capture topic information which is usually important to text retrieval task. In the future, to better improve the retrieval

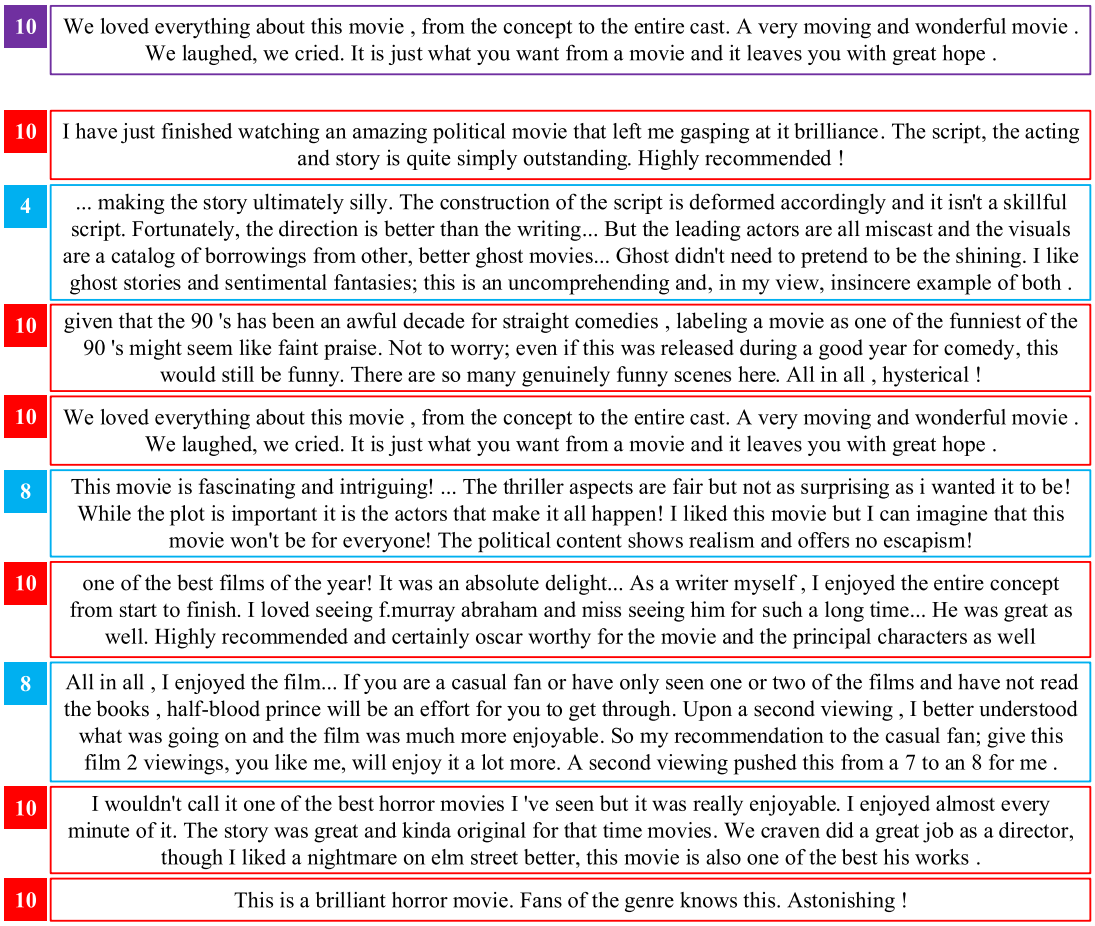

Fig. 5. Top retrieved 9 movie reviews returned by our proposed DSenH on a manually created query set based on the IMDB dataset. The review underlined in pink on the first row is the query sample. From top to bottom are the retrieved reviews when 48-bit binary codes are used for search. Note that the digits before reviews are the star labels. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

performance, we will not only encode the sentiment but also encode the topic information into the hash codes.

Acknowledgments

We are grateful to the anonymous reviewers for their helpful and thought-provoking suggestions and feedbacks. This work was supported in part by the National Natural Science Foundation of China under grants No. 61232004, No. 61502189 and No. 61672254 , and the National Key Research and Development Program of China No. 2016YFB0800402.

References

[1] L. Atzori, A. Iera, G. Morabito, The internet of things: A survey, Comput. Netw. 54 (15) (2010) 2787–2805.
[2] A. Botta, W. de Donato, V. Persico, A. Pescapè, Integration of cloud computing and internet of things: A survey, Future Gener. Comput. Syst. 56 (2016) 684–700.
[3] Q. Wu, G. Ding, Y. Xu, S. Feng, Z. Du, J. Wang, K. Long, Cognitive internet of things: A new paradigm beyond connection, IEEE Internet Things J. 1 (2) (2014) 129–143.
[4] X. Xu, L. He, H. Lu, A. Shimada, R. Taniguchi, Non-Linear matrix completion for social image tagging, IEEE Access 5 (2017) 6688–6696.
[5] X. Xu, L. He, A. Shimada, R. Taniguchi, H. Lu, Learning unified binary codes for cross-modal retrieval via latent semantic hashing, Neurocomputing 213 (2016) 191–203.
[6] A. Gionis, P. Indyk, R. Motwani, Similarity search in high dimensions via hashing, in: VLDB, 1999, pp. 518–529.
[7] A. Andoni, P. Indyk, Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions, in: FOCS, 2006, pp. 459–468.
[8] M. Raginsky, S. Lazebnik, Locality-sensitive binary codes from shift-invariant kernels, in: NIPS, 2009, pp. 1509–1517.
[9] B. Kulis, K. Grauman, Kernelized locality-sensitive hashing for scalable image search, in: ICCV, 2009, pp. 2130–2137.
[10] Y. Weiss, A. Torralba, R. Fergus, Spectral hashing, in: NIPS, 2008, pp. 1753–1760.
[11] B. Kulis, K. Grauman, Kernelized locality-sensitive hashing, IEEE Trans. Pattern Anal. Mach. Intell. 34 (6) (2012) 1092–1104.
[12] D. Zhang, J. Wang, D. Cai, J. Lu, Self-taught hashing for fast similarity search, in: SIGIR, 2010, pp. 18–25.
[13] W. Liu, J. Wang, S. Kumar, S. Chang, Hashing with graphs, in: ICML, 2011, pp. 1–8.
[14] Y. Gong, S. Lazebnik, A. Gordo, F. Perronnin, Iterative quantization: a procrustean approach to learning binary codes for large-scale image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 35 (12) (2013) 2916–2929.
[15] W. Liu, J. Wang, R. Ji, Y. Jiang, S. Chang, Supervised hashing with kernels, in: CVPR, 2012, pp. 2074–2081.
[16] B. Kulis, T. Darrell, Learning to hash with binary reconstructive embeddings, in: NIPS, 2009, pp. 1042–1050.
[17] J. Wang, J. Yang, K. Yu, F. Lv, T.S. Huang, Y. Gong, Locality-constrained linear coding for image classification, in: CVPR, 2010, pp. 3360–3367.
[18] M. Norouzi, D.J. Fleet, Minimal loss hashing for compact binary codes, in: ICML, 2011, pp. 353–360.
[19] G. Lin, C. Shen, D. Suter, A. van den Hengel, A general two-step approach to learning-based hashing, in: ICCV, 2013, pp. 2552–2559.
[20] X. Shi, F. Xing, K. Xu, M. Sapkota, L. Yang, Asymmetric discrete graph hashing, in: AAAI, 2017, pp. 2541–2547.
[21] D. Zhai, X. Liu, X. Ji, D. Zhao, S. Satoh, W. Gao, Supervised distributed hashing for large-scale multimedia retrieval, IEEE Trans. Multimed. (2017).
[22] R. Xia, Y. Pan, H. Lai, C. Liu, S. Yan, Supervised hashing for image retrieval via image representation learning, in: AAAI, 2014, pp. 2156–2162.
[23] K. Lin, H. Yang, J. Hsiao, C. Chen, Deep learning of binary hash codes for fast image retrieval, in: CVPR, 2015, pp. 27–35.

[24] V.E. Liong, J. Lu, G. Wang, P. Moulin, J. Zhou, Deep hashing for compact binary codes learning, in: CVPR, 2015, pp. 2475–2483.
[25] K. Zhou, Y. Liu, J. Song, L. Yan, F. Zou, F. Shen, Deep self-taught hashing for image retrieval, in: ACMMM, 2015, pp. 1215–1218.
[26] H.-F. Yang, K. Lin, C.-S. Chen, Supervised learning of semantics-preserving hash via deep convolutional neural networks, IEEE Trans. Pattern Anal. Mach. Intell. (2017).
[27] J. Song, T. He, L. Gao, X. Xu, H.T. Shen, Deep region hashing for efficient large-scale instance search from images, 2017. CoRRabs/1708.04150.
[28] J. Song, Binary generative adversarial networks for image retrieval, 2017. arXiv:1708.04150.
[29] J. Xu, P. Wang, G. Tian, B. Xu, J. Zhao, F. Wang, H. Hao, Convolutional neural networks for text hashing, in: IJCAI, 2015, pp. 1369–1375.
[30] S. Chaidaroon, Y. Fang, Variational deep semantic hashing for text documents, in: SIGIR, 2017, pp. 75–84.
[31] M. Belkin, P. Niyogi, Laplacian eigenmaps and spectral techniques for embedding and clustering, in: NIPS, 2001, pp. 585–591.
[32] Y. Weiss, R. Fergus, A. Torralba, Multidimensional spectral hashing, in: ECCV, 2012, pp. 340–353.
[33] M. Datar, N. Immorlica, P. Indyk, V.S. Mirrokni, Locality-sensitive hashing scheme based on p-stable distributions, in: Proceedings of the 20th ACM Symposium on Computational Geometry, 2004, pp. 253–262.
[34] Y. Kim, Convolutional neural networks for sentence classification, in: EMNLP, 2014, pp. 1746–1751.
[35] K.S. Tai, R. Socher, C.D. Manning, Improved semantic representations from tree-structured long short-term memory networks, in: ACL, 2015, pp. 1556–1566.
[36] J. Xu, D. Chen, X. Qiu, X. Huang, Cached long short-term memory neural networks for document-level sentiment classification, in: EMNLP, 2016, pp. 1660–1669.
[37] P. Bhatia, Y. Ji, J. Eisenstein, Better document-level sentiment analysis from rst Discourse Parsing, in: EMNLP, 2015, pp. 2212–2218.
[38] D. Tang, B. Qin, T. Liu, Document modeling with gated recurrent neural network for sentiment classification, in: EMNLP, 2015, pp. 1422–1432.
[39] V. Mnih, N. Heess, A. Graves, K. Kavukcuoglu, Recurrent models of visual attention, in: NIPS, 2014, pp. 2204–2212.
[40] K. Gregor, I. Danihelka, A. Graves, D.J. Rezende, D. Wierstra, DRAW: A recurrent neural network for image generation, in: ICML, 2015, pp. 1462–1471.
[41] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2014. arXiv:1409.0473.
[42] T. Luong, H. Pham, C.D. Manning, Effective approaches to attention-based neural machine translation, in: EMNLP, 2015, pp. 1412–1421.
[43] K. Xu, J. Ba, R. Kiros, K. Cho, A.C. Courville, R. Salakhutdinov, R.S. Zemel, Y. Bengio, Show, attend and tell: Neural image caption generation with visual attention, in: ICML, 2015, pp. 2048–2057.
[44] T. Rocktäschel, E. Grefenstette, K.M. Hermann, T. Kociský, P. Blunsom, Reasoning about entailment with neural attention, 2015. arXiv:1509.06664.
[45] Z. Yang, D. Yang, C. Dyer, X. He, A.J. Smola, E.H. Hovy, Hierarchical attention networks for document classification, in: NAACL, 2016, pp. 1480–1489.
[46] D. Tang, B. Qin, T. Liu, Learning semantic representations of users and products for document level sentiment classification, in: ACL, 2015, pp. 1014–1023.
[47] H. Chen, M. Sun, C. Tu, Y. Lin, Z. Liu, Neural sentiment classification with user and product attention, in: EMNLP, 2016, pp. 1650–1659.
[48] T. Mikolov, I. Sutskever, K. Chen, G.S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: NIPS, 2013, pp. 3111–3119.
[49] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[50] H. Lai, Y. Pan, Y. Liu, S. Yan, Simultaneous feature learning and hash coding with deep neural networks, in: CVPR, 2015, pp. 3270–3278.

Ke Zhou received the B.E., M.E., and Ph.D. degrees in computer science and technology from Huazhong University of Science and Technology (HUST), China, in 1996, 1999, and 2003, respectively. He is a full professor of the School of Computer Science and Technology, HUST. His main research interests include computer architecture, cloud storage, parallel I/O and storage security. He has more than 50 publications in journals and international conferences, including Performance Evaluation, FAST, MSST, ACM MM, SYSTOR, MASCOTS and ICC. He is a member of IEEE.

Jiangfeng Zeng is currently pursuing the Ph.D. degree at Wuhan National Laboratory for Optoelectronics, Huazhong University of Science and Technology, China.

His current research interests include machine learning, natural language processing, sentiment analysis, deep hashing and image generation.

Yu Liu received the B.S. degree and the M.S. degree from Computer Science and Technology, Wuhan Institute of Technology, Wuhan, China, in 2008 and 2012 respectively, and the Ph.D. degree in computer science from HUST in 2017. Currently, he is a postdoctoral researcher in the Dept. of Software Engineering, University of Science and Technology (HUST), China. His current research interests include machine learning, large-scale multimedia search, big data and storage etc.

Fuhao Zou received B.E. degree in computer science from Huazhong Normal University, Wuhan, Hubei, China, in 1998. And he received M.S. and Ph.D. in computer science and technology from Huazhong University of Science and Technology (HUST), Wuhan, Hubei, China, in 2003 and 2006. Currently, he is an associate professor with the college of computer science and technology, HUST. His research interests include machine learning, multimedia understanding and analysis, big data analysis, semantic based storage, and cloud storage. He is senior member of China Computer Federation (CCF) and member of IEEE, ACM.