Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence

Probabilistic Attributed Hashing

Mingdong Ou[1], **Peng Cui**[1], **Jun Wang**[2], **Fei Wang**[3], **Wenwu Zhu**[1]
[1]Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University. Beijing, China
[2]Data Science, Alibaba Group, Seattle, WA, USA.
[3]Department of Computer Science and Engineering, University of Connecticut. Storrs, CT. USA.
oumingdong@gmail.com, cuip@tsinghua.edu.cn, jwang@ee.columbia.edu, feiwang03@gmail.com, wwzhu@tsinghua.edu.cn

Abstract

Due to the simplicity and efficiency, many hashing methods have recently been developed for large-scale similarity search. Most of the existing hashing methods focus on mapping low-level features to binary codes, but neglect attributes that are commonly associated with data samples. Attribute data, such as image tag, product brand, and user profile, can represent human recognition better than low-level features. However, attributes have specific characteristics, including high-dimensional, sparse and categorical properties, which is hardly leveraged into the existing hashing learning frameworks. In this paper, we propose a hashing learning framework, `Probabilistic Attributed Hashing (PAH)`, to integrate attributes with low-level features. The connections between attributes and low-level features are built through sharing a common set of latent binary variables, i.e. hash codes, through which attributes and features can complement each other. Finally, we develop an efficient iterative learning algorithm, which is generally feasible for large-scale applications. Extensive experiments and comparison study are conducted on two public datasets, i.e., DBLP and NUS-WIDE. The results clearly demonstrate that the proposed `PAH` method substantially outperforms the peer methods.

Introduction

Attributes, which are often derived from human knowledge, represent characteristics of entities (e.g. users, images, etc.) in different aspects. From the view point of computational intelligence, attributes are imperative to improve data usability through giving a new view from human cognition. Thus, attributes have been ubiquitously used in various application scenarios, such as image and video tags in social media platform, product attributes (e.g. brands, materials etc.) in e-commerce, user profiles (e.g. age, gender etc.) in social networks and so on. These attributes provide an effective and efficient way for data management and organization. More importantly, users' interaction behaviors on entities show obvious patterns over entity attributes. For example, people tend to purchase products of a specific brand, like images and videos with a specific group of tags, and read papers written by a specific group of researchers. This property sheds lights on similarity search, where low-level features alone perform poorly. How to integrate attributes with low-level features for large-scale similarity search is of paramount significance for information retrieval, recommendation, and other real-world applications.

Hashing is a class of efficient methods for large-scale similarity search (Indyk and Motwani 1998; Weiss, Torralba, and Fergus 2008; Wang, Kumar, and Chang 2012; Liu et al. 2011). Most of the existing methods are designed to map low-level features to binary code space, namely Hamming space. Typically, attribute data has specific characteristics, such as high-dimensional, sparse, and categorical properties. For instance, in the public image dataset NUS-WIDE (Chua et al. 2009), a small subset of `Flickr`, there are over $400,000$ unique textual tags and each image is associated with just 18 tags on average. However, most of the existing hashing learning methods are more feasible for dense and continuous feature representations. Specifically, many of the linear projection methods rely on the estimation of a data covariance matrix for finding optimal projections (Weiss, Torralba, and Fergus 2008; Wang, Kumar, and Chang 2012; Gong and Lazebnik 2011; Strecha et al. 2012), but it is difficult to estimate the data covariance for extremely high-dimensional and sparse attribute data due to the well-known sample complexity issue. In addition, some existing supervised and semi-supervised learning frameworks for hash function design require acquiring a set of pairwise labels that can often be estimated by computing the amounts of commonly shared semantic tags (Zhang et al. 2010; Liu et al. 2011; Wang, Kumar, and Chang 2012). Due to extremely sparse attributes, such estimation for pairwise similarity will be much less reliable. Therefore, it remains a challenging issue to effectively leverage the attribute data into the hash function design.

In order to incorporate attributes to design more effective hash function, we need to address the following three challenges. First, how to exploit the complementary relationship between attributes and low-level features? Low-level features are often less descriptive but easily available. Although more descriptive, user-generated attributes often tend to be incomplete. Thus, the resulted hashing can expect to benefit much from the complementary properties of attributes and low-level features. Second, how to accommodate the differ-

ent characteristics of attributes and features? Since attributes are often in discrete and categorical space and features are often in continuous space, it requires different assumptions and hypothesis of the data. Third, how to achieve a scalable solution for learning hash functions? As attributes are high-dimensional and sparse, the hash models need to be learned on large-scale database in order to receive sufficient semantic information and avoid overfitting.

To address the above challenges, we propose a probabilistic generative model, `Probabilistic Attributed Hashing (PAH)`. Attributes and low-level features are both generated from a set of latent binary variables, i.e., hash codes. Thus, they can interact with each other through these latent variables. To cope with different data characteristics, we design two types of probabilistic generative processes for attributes and features, respectively. First, for attribute data, we employ a novel generative process based on multinomial mixture distributions, which can be easily adapted to the characteristics of attributes, and extract latent attributes clusters. Then, for low-level features, we use a Gaussian generative process to approximate the distribution of continuous low-level features. Besides, we develop an efficient iterative learning algorithm with linear time complexity for each updating process, and the learning of `PAH` is fairly scalable for large-scale application. With `PAH`, we can incorporate attributes into hashing, and generate more descriptive hash codes. We perform extensive experiments and compare with several representative hashing methods on two public datasets, DBLP[1] and NUS-WIDE. The results clearly demonstrate that the proposed `PAH` method outperforms the peer methods.

Related Work

Here we will briefly survey attribute based learning methods and some representative hash function design approaches.

Learning with Attributes

As attributes can be more descriptive, many researchers analyse and exploit them from various perspectives. In recommendation systems, Jiang et al. (2012a; 2012b) incorporate both friend relationships and tags into recommendation algorithms to model the similarity between users. In document representation, El-Arini et al. (2013) propose to model documents by attributes of readers. Kim et al. (2012) exploits entities in documents to build more expressive, entity-based topic model. Tang et al. (2013) uses different types of context information to improve the performance of topic modeling on sparse user-generated data. In computer vision, the attributes are often regarded as multiple semantic labels. Hence, the researchers focus on learning attribute representation of images that can further be used in object description (Farhadi et al. 2009), image retrieval (Zhang et al. 2013; Yu et al. 2012; Douze, Ramisa, and Schmid 2011), and object recognition (Lampert, Nickisch, and Harmeling 2009).

[1]http://www.informatik.uni-trier.de/~ley/db/

Hashing

Hashing provides an efficient solution for approximate nearest neighbor (ANN) search in large-scale high dimensional space. Many hashing methods are designed to improve the search accuracy for a single type of features (Indyk and Motwani 1998; Weiss, Torralba, and Fergus 2008; Salakhutdinov and Hinton 2009; Liu et al. 2011; Gong and Lazebnik 2011; Heo et al. 2012). For example, `Locality sensitive hashing (LSH)` (Indyk and Motwani 1998) generates hash codes by random projections. `Spectral hashing` (Weiss, Torralba, and Fergus 2008) learns hash functions from the data distribution in feature space.

In addition, various kinds of extra information are exploited to further improve the accuracy of hashing based ANN search. As supervision information is available in many scenarios, supervised (or semi-supervised) hashing methods were proposed (Kulis, Jain, and Grauman 2009; Liu et al. 2012; Norouzi, Fleet, and Salakhutdinov 2012; Torralba, Fergus, and Weiss 2008; Wang, Kumar, and Chang 2012). For example, `Semi-supervised hashing(SSH)` (Wang, Kumar, and Chang 2012) is proposed to learn accurate and balanced hash codes when supervision information is limited. `KSH` (Liu et al. 2012) introduces kernelization into hash function design with affordable training cost.

Realizing that multiple types of features are also ubiquitous (e.g. multiple image visual features), multi-modal hashing methods have been recently invented (Bronstein et al. 2010; Kumar and Udupa 2011; Song et al. 2011; Zhen and Yeung 2012; Zhang, Wang, and Si 2011; Ou et al. 2013). Briefly, multi-modal hashing methods leverage complementary features to generate more representative binary codes. For example, Zhen et al. (2012) propose a probabilistic model which preserves inter-modal and intra-modal similarity simultaneously in the projected Hamming space. Zhang et al. (2011) design composite hashing with multiple source data (CHMS). Although these multi-modal hashing methods can deal with multiple types of features, they rely on a common model for all types of features. However, due to the significantly different properties of the low-level features and the attributes, the aforementioned multi-modal hashing methods often produce poor performance if directly applied to our setting. Finally, Wang et al. (2013) propose a semantic hashing method (SHTTM) using document tags. But, it uses document tags only in training procedure and neglects the sparsity of tags which makes SHTTM not scalable for high-dimensional attributes.

Framework of Probabilistic Attributed Hashing

In this section, we first give the formal problem statement of Probabilistic Attributed Hashing (`PAH`), and then describe the details of the formulation and solution.

Notations

Attributes and features of entities are the inputs of `PAH`. Let $\mathbf{E} = \{e_1, e_2, ..., e_N\}$ be the set of entities, where N is the

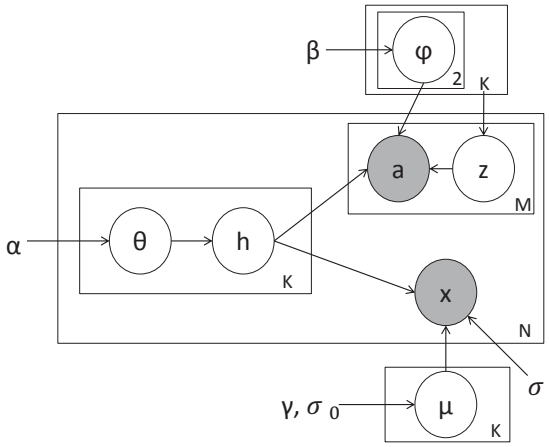


Figure 1: The graphical model of probabilistic attributed hashing. Algorithm 1 presents the corresponding generative process.

number of entities. Each entity e_i has an attribute vector $\mathbf{a}_i = \{a_{i1}, a_{i2}, \cdots, a_{iM_i}\}$, where M_i is the number of attributes tagged on e_i. All the attributes are from a attribute vocabulary indexed by $\{1, 2, \cdots, L_a\}$, where L_a is the size of the attribute vocabulary. Let $\mathbf{A} = \{\mathbf{a}_1, \mathbf{a}_2, \cdots, \mathbf{a}_N\}$ be the set of attribute vectors. We denote $\mathbf{X} \in \mathbb{R}^{L_f \times N}$ as the feature matrix, where the i-th column vector $\mathbf{x}_i$ is the feature vector of the entity e_i. The dimension of features is given as L_f, respectively. PAH learns hash codes using both attributes and features. Assume that $\mathbf{H} \in \{0, 1\}^{K \times N}$ is a binary matrix and its i-th column $\mathbf{h}_i$ is the hash code of e_i and each hash code consists of K bits. Let $\mathbf{Y} \in \{1, -1\}^{K \times N}$ be the variant of $\mathbf{H}$, where $y_{ki} = 2 * h_{ki} - 1$.

Model Formulation

As shown in Figure 1 (Algorithm 1 presents the corresponding generative process), PAH is a probabilistic generative model. As attributes and low-level features are different representations of common entities and complementary to each other, we use common binary hidden variables (h in Figure 1) to integrate the information from the two representations. In other words, attributes and low-level features are both generated from common hash codes. The generation mechanisms tend to be fairly different due to different characteristics of the attributes $\mathbf{A}$ and the features $\mathbf{X}$. With the above framework, attributes and low-level features are linked up, and can transfer information to each other. Below we will present each process in detail.

First, we construct a binary generative model for observed attributes. As Latent Dirichlet Allocation (LDA) (Blei, Ng, and Jordan 2003) has been a powerful tool for extracting latent semantic topics from high-dimensional and sparse documents, we adopt a generative structure like LDA. Unlike the standard LDA, we propose a novel binary generative process to fit the discrete property of hash bits and attributes. Each value of each bit (i.e. 0 or 1) corresponds to a cluster of attributes, and we represent it by a multinomial distribution on attributes, i.e. $p(a_{im}|\mathbf{h}_i, \varphi, z_{im}) = \varphi_{h_{i,z_{im}}, z_{im}, a_{im}}$. Here, $\varphi \in \mathbb{R}^{2 \times K \times L_a}$ is a tensor denoting multinomial distributions, $\sum_l \varphi_{b,k,l} = 1$. Let $\varphi_{b,k} = \{\varphi_{b,k,1}, \varphi_{b,k,2}, \cdots, \varphi_{b,k,L_a}\}$. z_{im} determines that a_{im} generates from the multinomial distributions of z_{im}-th hash bit. To balance the hash bits, we generate z_{im} uniformly, $z_{im} \sim \mathrm{Unif}(1, 2, \cdots, K)$. The hash bits are generated from Bernoulli distributions, $h_{ki} \sim \mathrm{Bern}(\theta_{ki})$. Hence, each hash code represents a mixture of hash bits:

$$p(a_{im}|\mathbf{h}_i, \varphi) = \frac{1}{K} \sum_{z_{im}=1}^{K} \varphi_{h_{i,z_{im}}, z_{im}, a_{im}} \tag{1}$$

Thus, like LDA, attributes can be clustered through the co-occurrence in attribute set of entities. Such a cluster strategy using local relationship can avoid evaluating similarity (or covariance) on sparse attributes which is not accurate as mentioned in Introduction.

Besides, we choose the following two conjugate prior distributions to generate φ and θ:

$$\varphi \sim \mathrm{Dir}(\beta), \;\; \theta \sim \mathrm{Beta}(\alpha), \tag{2}$$

where β and α are the hyper-parameters. Here, $\mathrm{Dir}(x)$ and $\mathrm{Beta}(x)$ represent Dirichlet and beta distributions.

Then, for low-level features, we adopt a Gaussian generative model. We assume that the feature data $\mathbf{X}$ is generated from continuous Gaussian distributions. Take i-th entity as example. Each value of each hash bit corresponds to a sub-center (i.e. $y_{ki}\mu_k$), and each hash code corresponds to a specific Gaussian center which is a mixture of sub-centers, $\sum_k^K y_{ki}\mu_k$. Thus, similar hash codes will generate similar low-level features.

$$\mathbf{x}_i \sim \mathrm{N}(\sum_k^K y_{ki}\mu_k, \sigma * \mathbf{I}) \tag{3}$$

Algorithm 1 Probabilistic Attributed Hashing

for all hash bit h **do**
  for all bit value $\{0,1\}$ **do**
    Draw φ from Dirichlet distribution $\mathrm{Dir}(\beta)$
  end for
  Draw μ from Gaussian distribution $\mathrm{N}(\gamma, \sigma_0 * \mathbf{I})$
end for
for all entity e **do**
  for all hash bit h **do**
    Draw θ from Beta distribution $\mathrm{Beta}(\alpha)$
    Draw h from Bernoulli distribution $\mathrm{Bern}(\theta)$
  end for
  for all attribute a **do**
    Draw z from uniform distribution $\mathrm{Unif}(1, 2, \cdots, K)$
    Draw a from multinomial distribution $\mathrm{Multi}(\varphi_{h_z, z})$
  end for
  Draw x from Gaussian distribution $\mathrm{N}(\sum_k^K y_k\mu_k, \sigma * \mathbf{I})$
end for

where $\mathbf{I}$ is an identity matrix and σ is the parameter of the covariance.

Besides, we adopt the conjugate prior distribution below to generate μ.

$$\mu_k \sim \mathrm{N}(\gamma, \sigma_0 * \mathbf{I}), \tag{4}$$

where γ and σ_0 are two hyper-parameters.

Iterative learning algorithm

To achieve optimal latent variables, i.e. $\mathbf{H}$, $\varphi \in [0,1]^{2\times K\times L_a}$, and $\mu \in \mathbb{R}^{L_f\times K}$, we use MAP (Maximum a posteriori) to conduct model estimations. The posterior distribution is given as:

$$p(\mathbf{H},\varphi,\mu|\mathbf{A},\mathbf{X},\alpha,\beta,\gamma) \propto p(\mathbf{A}|\mathbf{H},\varphi) * p(\mathbf{X}|\mathbf{H},\mu) * p(\mathbf{H}|\alpha) * p(\varphi|\beta) * p(\mu|\gamma) \tag{5}$$

As it is intractable to perform the learning process simultaneously, we propose to learn $\mathbf{H}$, φ, μ in an iterative way.

We update hash codes $\mathbf{H}$ bit by bit, i.e. we update each bit h_{ik} by fixing other hash bits. Formally, we need to maximize the following posterior distribution of h_{ik}

$$p(h_{ik}|\mathbf{h}_{-ik},\mathbf{x}_i,\mathbf{a}_i,\varphi,\alpha,\mu) = \frac{1}{Z}p(\mathbf{a}_i|\mathbf{h}_i,\varphi) * p(\mathbf{x}_i|\mathbf{h}_i,\mu) * p(h_{ik}|\alpha) \tag{6}$$

with

$$p(\mathbf{a}_i|\mathbf{h}_i,\varphi) = \prod_m^{M_i} p(a_{im}|\mathbf{h}_i,\varphi)$$

$$p(h_{ik}|\alpha) = \int p(h_{ik}|\theta)p(\theta|\alpha)d\theta = \frac{\alpha_{h_{ik}}}{\alpha_0+\alpha_1}.$$

Here $\mathbf{h}_{-ik}$ is the hash bits in $\mathbf{h}_i$ except the bit h_{ik} and Z are normalization constants.

Since h_{ik} is binary, we can get optimal h_{ik} by comparing the two posterior probabilities of h_{ik}:

$$h_{ik} = \mathrm{sgn}(\log\frac{p(h_{ik}=1|\mathbf{h}_{-ik},\mathbf{x}_i,\mathbf{a}_i,\varphi,\alpha,\mu)}{p(h_{ik}=0|\mathbf{h}_{-ik},\mathbf{x}_i,\mathbf{a}_i,\varphi,\alpha,\mu)}), \tag{7}$$

where $\mathrm{sgn}(\cdot)$ is the sign function. Note that the hash codes of queries are also computed by this algorithm with the learned parameters φ and μ.

We use the EM (Expectation Maximization) algorithm to learn φ through solving the following problem:

$$\min \quad -\sum_i^N\sum_m^{M_i}\sum_{z_{im}}^K q(z_{im}|\mathbf{h}_i,\varphi)\log\frac{p(a_{im},z_{im}|\mathbf{h}_i,\varphi)}{q(z_{im}|\mathbf{h}_i,\varphi)} - \sum_{b=0}^1\sum_k^K \beta^\top \log\varphi_{b,k}. \tag{8}$$

E step: we calculate the posterior distribution as

$$\begin{aligned} q(z_{im}|\mathbf{h}_i,\varphi) &= p(z_{im}|\mathbf{h}_i,\varphi,a_{im}) \\ &= \frac{p(a_{im}|z_{im},\mathbf{h}_i,\varphi)p(z_{im})}{p(a_{im}|\mathbf{h}_i,\varphi)} \\ &= \frac{1}{C_{z_{im}}}\varphi_{h_{i,z_{im}},z_{im},a_{im}} \end{aligned}$$

where $C_{z_{im}}$ is a constant.

M step: we maximize φ with $q(z_{im}|\mathbf{h}_i,\varphi)$ fixed. To handle the constraint $\sum_l^{L_a}\varphi_{b,k,l}=1$, we use the Lagrange multiplier and maximize the following objective:

$$\max \sum_i^N\sum_m^{M_i}\sum_{z_{im}}^K q(z_{im}|\mathbf{h}_i,\varphi)\log\frac{p(a_{im},z_{im}|\mathbf{h}_i,\varphi)}{q(z_{im}|\mathbf{h}_i,\varphi)} + \sum_{b=0}^1\sum_k^K \beta^\top\log\varphi_{b,k} + \lambda_{b,k}(\sum_l^{L_a}\varphi_{b,k,l}-1).$$

We can compute the optimal solution as:

$$\varphi_{b,k,l} = \frac{\sum_{i\in\mathcal{I}}\sum_{m\in\mathcal{M}} q(z_{im}=k|\mathbf{h}_i,\varphi)+\beta_l}{\sum_{i\in\mathcal{I}}\sum_m^{M_i} q(z_{im}=k|\mathbf{h}_i,\varphi)+\sum_l\beta_l}, \tag{9}$$

where $\mathcal{I}=\{i|h_{ik}=b\}$, $\mathcal{M}=\{m|a_{im}=l\}$.

Similar to the learning of φ, we estimate the parameter μ through maximizing the following posterior distribution:

$$\log(p(\mu|\mathbf{X},\mathbf{H},\gamma)) = -\sum_i^N\|\mathbf{x}_i-\mu\mathbf{h}_i\|_F^2 - \sum_k^K\|\mu_k-\gamma\|_F^2 + C_\mu \tag{10}$$

where C_μ is a constant. Then we can get the optimal estimation of μ as

$$\mu = (\gamma\mathbf{1}^\top + \mathbf{X}\mathbf{H}^\top)(\mathbf{I}+\mathbf{H}\mathbf{H}^\top)^{-1}. \tag{11}$$

Complexity Analysis

For the learning process of $\mathbf{H}$, we need to calculate the probability of each hash bit. The time complexity is $O(K*(C_a+N*L_f))$, where C_a is the total counts of all attributes, i.e. the number of nonzero elements in $\mathbf{A}$. For the learning of φ, we only need to count the frequency of attributes for each latent attribute and approximate the probability q, resulting in the time complexity as $O(K*C_a)$. For the learning of μ, the time complexity is $O(N*K*L_f+K^3)$. In summary, the time complexity of each iteration is $O(K*(C_a+N*L_f+K^2))$. Note that the time complexity is linear with respect to the number of entities. Since the length of compact hash codes (the value of K) is often set to be small, the training procedure of the proposed `PAH` method is fairly efficient in practice. Moreover, as the hash codes are updated independently and the other latent variables (i.e. φ and μ) are updated based on some simple statistics, we can implement parallel `PAH` easily.

Experiments

In this section, we describe the experimental settings and report the results.

Datasets

We perform the experiments and comparison using two popular benchmark datasets, i.e. the DBLP and the NUS-WIDE dataset (Chua et al. 2009). NUS-WIDE is an image dataset crawled from `Flickr`, which includes about $260,000$ images of 81 concept categories. We select the top 10 popular

Table 1: Selected research fields and corresponding conferences

Field	Conference
Database	ICDE, VLDB, SIGMOD, PODS, EDBT
Data Mining	KDD, ICDM, SDM, PKDD, PAKDD
Artificial Intelligence	IJCAI, AAAI
Information Retrieval	SIGIR, ECIR
Computer Vision	CVPR
Machine Learning	ICML, ECML
Algorithms & Theory	STOC, FOCS, SODA, COLT
Natural Language Processing	ACL, ANLP, COLING
Bioinformatics	ISMB, RECOMB
Networking	SIGCOMM, MOBICOM, INFOCOM
Operating Systems	SOSP, OSDI
Distributed & Parallel Computing	PODC, ICS

concepts, and there are about $180,000$ images. All the images are described by user-generated tags. The $5,018$ tags selected by NUS-WIDE are used as attributes, and 500-dimension Bag-of-Visual-Word (BOW) of SIFT as features. We randomly select $10,000$ images as our training set, and $5,000$ images as our test set for the comparison with other methods. To test the efficiency of PAH, we randomly select multiple training sets, where the number of images ranges from 100 to $100,000$. We use the concepts as the ground truth, i.e. two images are regarded similar when they share common concepts, otherwise, they are dissimilar.

The DBLP dataset consists of the bibliographic information of the computer science community. We select 12 fields and 33 corresponding conferences (Table 1), which include a total of $64,256$ papers. We keep the authors whose number of papers is not less than 5, then get $3,527$ unique authors, and about $20,000$ papers. For each paper, we use the authors as our attributes, and its distributions over 100 LDA topics (extracted from the titles and abstracts) as features. We randomly select $10,000$ papers as our training set, and another $10,000$ papers as queries for the comparison with other methods. We use the research field labels as the ground truth, i.e. two papers are regarded as similar when they are published in a common field; otherwise, they are dissimilar.

Experiment Settings

We compare PAH with MLBE (Zhen and Yeung 2012), CVH (Kumar and Udupa 2011), CHMS (Zhang, Wang, and Si 2011), SHTTM (Wang, Zhang, and Si 2013), AGH (Liu et al. 2011), SH (Weiss, Torralba, and Fergus 2008). Note that MLBE, CVH, CHMS are multi-modal hashing methods that treat features and attributes equally. SHTTM uses attributes as supervision information, and only low-level features are used as input of hash functions. We treat BOW of SIFT in NUS-WIDE as document for SHTTM. In addition, we include two simplified versions of the PAH methods, i.e., the PAH_NF and the PAH_NA, where PAH_NF exploits attributes only (omits the features) and PAH_NA exploits features only (omits the attributes). Since AGH and SH are two single feature based methods, we concatenate attributes and features into a single feature for both of them.

Given a query, the experimental evaluation considers the most similar (based on Hamming distance) entities. Here we measure the search quality using the MAP (Mean Average Precision) of top 500 returned results and the Precision-Recall curve, where precision-recall pairs are obtained by changing the search space range from Hamming radius 0 to K. For setting the hyper-parameters in PAH, we use grid search to get the optimal hyper-parameters, and get $\alpha = \{0.1\}^{2\times 1}$, $\beta = \{0.01\}^{L_a\times 1}$, $\gamma = \{0\}^{L_f\times 1}$. Note that the parameters σ, σ_0 weigh the contributions from the feature data. Hence we get $\sigma = \sigma_0 = 1.0$ for the DBLP dataset, and $\sigma = \sigma_0 = 10^{-5}$ for the NUS-WIDE dataset, because the document features represented by the distributions of the hidden topics provide rich and reliable semantic meanings.

Finally, all the algorithms are implemented using Matlab (the codes of baselines are provided by their authors), and run experiments on a machine running Windows Server 2008 with 12 2.4GHz cores and 192GB memory.

Experimental Results

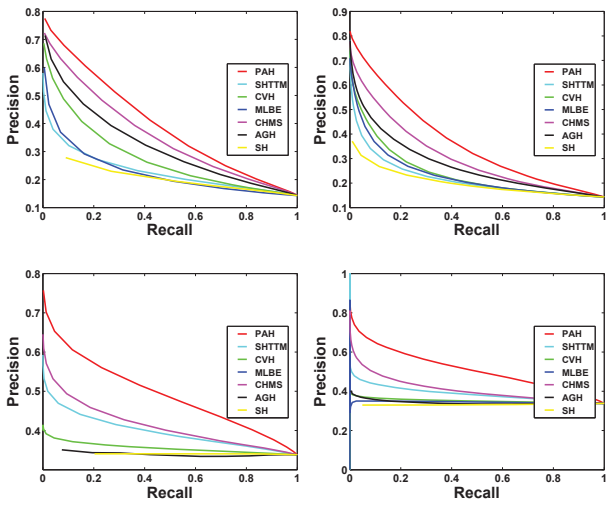


Figure 2: Comparison study of the Precision-Recall curves using different hashing methods. The top row shows the results on the DBLP data, and bottom row shows the results on the NUS-WIDE data. From left to right, the subfigures demonstrate the results using 16 and 32 hash bits, respectively.

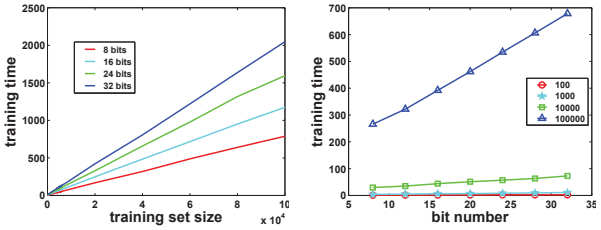
Figure 2 shows the precision-recall curves, where the PAH method achieves the best performance in all test settings. In Table 2, we show the performance measured by the MAP on the DBLP and the NUS-WIDE datasets. Compared with the best competing methods (except PAH_NF and PAH_NA), the proposed PAH achieves around $\sim 7\%$ relative improve-

Table 2: The MAPs on DBLP and NUS-WIDE when the number of hash bits varies. PAH_NF and PAH_NA are two variants of PAH, the others are competitive methods proposed in other papers.

Method	DBLP				NUS-WIDE			
	8 bits	16 bits	24 bits	32 bits	8 bits	16 bits	24 bits	32 bits
PAH	**0.530**	**0.563**	**0.575**	**0.573**	**0.566**	**0.597**	**0.613**	**0.625**
PAH_NF	0.509	0.533	0.537	0.532	0.563	0.597	0.609	0.619
PAH_NA	0.362	0.464	0.469	0.487	0.395	0.394	0.410	0.419
SHTTM	0.367	0.359	0.389	0.387	0.492	0.478	0.467	0.463
CVH	0.456	0.464	0.460	0.457	0.383	0.382	0.379	0.379
MLBE	0.256	0.267	0.270	0.250	0.350	0.350	0.350	0.350
CHMS	0.485	0.540	0.538	0.535	0.486	0.477	0.478	0.488
AGH	0.461	0.491	0.487	0.484	0.394	0.401	0.411	0.415
SH	0.299	0.347	0.361	0.366	0.352	0.355	0.355	0.357

ment on the DBLP, and $\sim$ 24% relative improvement on the NUS-WIDE. The prominent advantage of PAH is attributed to designing different generative processes to cope with the specific characteristics of features and attributes. In addition, the PAH method directly learns binary hash codes, and avoids the suboptimality from performing binarization as a postprocessing step. While the competing methods almost achieve the upper limit of performance when using 16-bit hash codes, the MAP of PAH keeps increasing when the number of hash bits increases. It indicates that our method is more powerful to prevent overfitting and has stronger generalization ability.

Since the PAH method outperforms both the PAH_NF and the PAH_NA on MAP, it clearly demonstrates that both attributes and features are complementary to each other, and a joint use of both data tends to give better performance. We also notice that PAH receives much higher improvement over PAH_NF on DBLP than that on the NUS-WIDE. Such a phenomenon is consistent with our initial hypothesis that the topic distribution based features on the DBLP data is a more powerful descriptor than the BOW representation of the NUS-WIDE data. In addition, the PAH_NF method works better than the PAH_NA method on both datasets, which indicates that attributes are more descriptive and are better representation for assessing the similarities.



(a) (b)

Figure 3: Training time of PAH on NUS-WIDE. The left figure shows the trend of training time when the size of training set varies (Different lines show the training time with different bit nubmers). The right figure shows the trend of training time when bit number varies (Different lines show the training time with different numbers of training examples).

Finally, we show the training time of PAH (Figure 3) to demonstrate the scalability in practice. Figure 3(a) shows that training time grows linearly with the size of training set. Figure 3(b) shows that training time grows almost linearly with respect to the number of hash bits when the training set is large (e.g. 10000 and 100000). Although the time complexity is $O(K * (C_a + N * L_f + K^2))$, the bit number K is often much smaller than the training set size in practice (e.g. $32^2 \ll 10000$), so that the training time is almost linear with respect to the number of hash bits. In addition, when the hash bit number is 32, each training example just costs about 20 milliseconds on average. So, PAH is feasible even for very large-scale applications.

Conclusion

In this paper, we have proposed a probabilistic generative model, namely Probabilistic Attributed Hashing (PAH), to integrate attributes and features for improving the accuracy of hashing. PAH employs different generative processes for attributes and features to capture their unique characteristics. We designed an efficient iterative learning algorithm where binary hash codes are learned directly. Empirical study on two distinct benchmark datasets has demonstrated the superiority of PAH, compared to several representative hashing methods.

As input data is becoming increasingly more diverse, our future work may focus on extending attributes to other complex types of data, such as relative attributes (Parikh and Grauman 2011). Besides, developing a supervised version of PAH would be another interesting direction to pursue.

Acknowledgement

This work was supported by National Program on Key Basic Research Project, No. 2015CB352300, No.2011CB302206; National Natural Science Foundation of China, No.61370022, No. 61472444 and No.61210008; International Science and Technology Cooperation Program of China, No. 2013DFG12870. Thanks for the support of NExT Research Center funded by MDA, Singapore, under the research grant, WBS:R-252-300-001-490 and the research fund of Tsinghua-Tencent Joint Laboratory for Internet Innovation Technology.

References

Blei, D. M.; Ng, A. Y.; and Jordan, M. I. 2003. Latent dirichlet allocation. *the Journal of machine Learning research* 3:993–1022.

Bronstein, M.; Bronstein, A.; Michel, F.; and Paragios, N. 2010. Data fusion through cross-modality metric learning using similarity-sensitive hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 3594–3601.

Chua, T.-S.; Tang, J.; Hong, R.; Li, H.; Luo, Z.; and Zheng, Y. 2009. Nus-wide: a real-world web image database from national university of singapore. In *Proceedings of the ACM International Conference on Image and Video Retrieval*, 48.

Douze, M.; Ramisa, A.; and Schmid, C. 2011. Combining attributes and fisher vectors for efficient image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 745–752. IEEE.

El-Arini, K.; Xu, M.; Fox, E. B.; and Guestrin, C. 2013. Representing documents through their readers. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '13, 14–22. ACM.

Farhadi, A.; Endres, I.; Hoiem, D.; and Forsyth, D. 2009. Describing objects by their attributes. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 1778–1785. IEEE.

Gong, Y., and Lazebnik, S. 2011. Iterative quantization: A procrustean approach to learning binary codes. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, 817–824. IEEE.

Heo, J.-P.; Lee, Y.; He, J.; Chang, S.-F.; and Yoon, S.-E. 2012. Spherical hashing. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2957–2964.

Indyk, P., and Motwani, R. 1998. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, 604–613.

Jiang, M.; Cui, P.; Liu, R.; Yang, Q.; Wang, F.; Zhu, W.; and Yang, S. 2012a. Social contextual recommendation. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 45–54. ACM.

Jiang, M.; Cui, P.; Wang, F.; Yang, Q.; Zhu, W.; and Yang, S. 2012b. Social recommendation across multiple relational domains. In *Proceedings of the 21st ACM international conference on Information and knowledge management*, 1422–1431. ACM.

Kim, H.; Sun, Y.; Hockenmaier, J.; and Han, J. 2012. Etm: Entity topic models for mining documents associated with entities. In *Proceedings of the 2012 IEEE 12th International Conference on Data Mining*, ICDM '12, 349–358. IEEE Computer Society.

Kulis, B.; Jain, P.; and Grauman, K. 2009. Fast similarity search for learned metrics. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 31(12):2143–2157.

Kumar, S., and Udupa, R. 2011. Learning hash functions for cross-view similarity search. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence*, 1360–1365.

Lampert, C. H.; Nickisch, H.; and Harmeling, S. 2009. Learning to detect unseen object classes by between-class attribute transfer. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, 951–958. IEEE.

Liu, W.; Wang, J.; Kumar, S.; and Chang, S.-F. 2011. Hashing with graphs. In *Proceedings of the 28th International Conference on Machine Learning*, 1–8.

Liu, W.; Wang, J.; Ji, R.; Jiang, Y.-G.; and Chang, S.-F. 2012. Supervised hashing with kernels. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2074–2081.

Norouzi, M.; Fleet, D.; and Salakhutdinov, R. 2012. Hamming distance metric learning. In *Advances in Neural Information Processing Systems*, 1070–1078.

Ou, M.; Cui, P.; Wang, F.; Wang, J.; Zhu, W.; and Yang, S. 2013. Comparing apples to oranges: a scalable solution with heterogeneous hashing. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 230–238. ACM.

Parikh, D., and Grauman, K. 2011. Relative attributes. In *Computer Vision (ICCV), 2011 IEEE International Conference on*, 503–510. IEEE.

Salakhutdinov, R., and Hinton, G. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50(7):969–978.

Song, J.; Yang, Y.; Huang, Z.; Shen, H. T.; and Hong, R. 2011. Multiple feature hashing for real-time large scale near-duplicate video retrieval. In *Proceedings of the 19th ACM international conference on Multimedia*, 423–432.

Strecha, C.; Bronstein, A. M.; Bronstein, M. M.; and Fua, P. 2012. Ldahash: Improved matching with smaller descriptors. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34(1):66–78.

Tang, J.; Zhang, M.; and Mei, Q. 2013. One theme in all views: modeling consensus topics in multiple contexts. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, 5–13. ACM.

Torralba, A.; Fergus, R.; and Weiss, Y. 2008. Small codes and large image databases for recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Wang, J.; Kumar, S.; and Chang, S.-F. 2012. Semi-supervised hashing for large-scale search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34(12):2393–2406.

Wang, Q.; Zhang, D.; and Si, L. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, 213–222. ACM.

Weiss, Y.; Torralba, A.; and Fergus, R. 2008. Spectral hashing. In *Advances in Neural Information Processing Systems*, 1753–1760.

Yu, F. X.; Ji, R.; Tsai, M.-H.; Ye, G.; and Chang, S.-F. 2012. Weak attributes for large-scale image retrieval. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, 2949–2956. IEEE.

Zhang, D.; Wang, J.; Cai, D.; and Lu, J. 2010. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, 18–25. ACM.

Zhang, H.; Zha, Z.-J.; Yang, Y.; Yan, S.; Gao, Y.; and Chua, T.-S. 2013. Attribute-augmented semantic hierarchy: towards bridging semantic gap and intention gap in image retrieval. In *Proceedings of the 21st ACM international conference on Multimedia*, 33–42. ACM.

Zhang, D.; Wang, F.; and Si, L. 2011. Composite hashing with multiple information sources. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval*, 225–234. ACM.

Zhen, Y., and Yeung, D. 2012. A probabilistic model for multimodal hash function learning. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, 940–948.