Learning Compact Hashing Codes for Efficient Tag Completion and Prediction

Qifan Wang, Lingyun Ruan, Zhiwei Zhang, and Luo Si
Computer Science Department
Purdue University
West Lafayette, IN 47907, US
{wang868,ruan1,zhan1187,lsi}@purdue.edu

ABSTRACT

Tags have been popularly utilized in many applications with image and text data for better managing, organizing and searching for useful information. Tag completion provides missing tag information for a set of existing images or text documents while tag prediction recommends tag information for any new image or text document. Valuable prior research has focused on improving the accuracy of tag completion and prediction, but limited research has been conducted for the efficiency issue in tag completion and prediction, which is a critical problem in many large scale real world applications.

This paper proposes a novel efficient Hashing approach for Tag Completion and Prediction (HashTCP). In particular, we construct compact hashing codes for both data examples and tags such that the observed tags are consistent with the constructed hashing codes and the similarities between data examples are also preserved. We then formulate the problem of learning binary hashing codes as a discrete optimization problem. An efficient coordinate descent method is developed as the optimization procedure for the relaxation problem. A novel binarization method based on orthogonal transformation is proposed to obtain the binary codes from the relaxed solution. Experimental results on four datasets demonstrate that the proposed approach can achieve similar or even better accuracy with state-of-the-art methods and can be much more efficient, which is important for large scale applications.

Categories and Subject Descriptors

I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Tag Completion, Hashing, Multi-Label Learning

CIKM'13, Oct. 27–Nov. 1, 2013, San Francisco, CA, USA.

http://dx.doi.org/10.1145/2505515.2505649.

1. INTRODUCTION

The purpose of data tagging, assigning tags or keywords to images, documents and video clips, is to benefit people for managing, organizing and searching desired information from various resources. Users can better categorize or search desired data based on the tags associated with them. Due to the rapid growth of the internet, a huge amount of data has been generated and users can only manually tag a very small portion of the data. Therefore, it is a challenging issue to complete missing tags for existing examples and predict tags for query examples on large scale data.

In the problems of tag completion and prediction, we are given a collection of data examples associated with a small number of tags. The goal is to complete missing tags for existing data examples, and to predict tags for unseen (query) examples. A large amount of research have been conducted on improving the accuracy of tag completion and prediction and generated promising results. However, most existing methods only focus on the effectiveness without paying much attention to efficiency. In many real world applications, the data size grows explosively and there are often a large number of possible tags and thus efficiency is a practical and critical issue. It is a practical and important research problem to design efficient methods for large scale tag completion and prediction.

This paper proposes a novel efficient approach for tag completion and prediction by designing compact binary hashing codes for both data examples and tags. In particular, each data example is represented by a C-bit binary code and each tag is also represented using a C-bit binary code. Our key idea of constructing the hashing codes is that *a tag should be assigned to a data example if the Hamming distance between their corresponding codes is small, and two similar data examples should also have similar codes.* We then formulate the problem of learning binary hashing codes as a discrete optimization problem by simultaneously ensuring the observed tags to be consistent with the constructed hashing codes and preserving the similarities between data examples.

2. RELATED WORK

This section reviews the related work in two areas: tag completion and prediction and learning hashing codes.

2.1 Tag Completion and Prediction

Many research on tag completion and prediction have been proposed in the past several years. Matrix completion [1, 2] methods can be applied to the tag completion task by

recovering the missing entries in the tag matrix. Cabral et al. [1] propose two convex algorithms for matrix completion based on a rank minimization criterion. Cai et al. [2] introduce an efficient singular value thresholding (SVT) algorithm for completing matrix. Tag completion and prediction can also be viewed as a multi-label learning problem that exploits the dependence among labels/tags, where each data example is associated with multiple labels/tags. Numerous work have been proposed on multi-label learning for automatic image/document annotation and classification [3, 9, 10, 6]. Desai et al. [6] propose a discriminative model for multi-label learning. Hariharan et al. [10] introduce a max-margin framework for large scale multi-label classification. A conditional dependency networks is utilized in [9] to structured multi-label learning. In work [3], Chen et al. propose an efficient multi-label classification method using hypergraph regularization.

Besides the multi-label learning methods, several recent machine learning approaches have been proposed to tag completion and prediction, including tag propagation [8], distance metric learning [14] and tag recommendation [7, 18]. Li et al. [14] propose a neighbor voting algorithm for social tagging which accurately and efficiently learns tag relevance by accumulating votes from visual neighbors. A tag propagation (TagProp) method has been proposed in [8] that propagates the label/tag information from the labeled examples to the unlabeled examples via a weighted nearest neighbor graph. More recently, Wu et al. [23] introduce a direct tag matrix completion algorithm by ensuring the completed tag matrix to be consistent with both the observed tags and the visual similarity.

Existing methods generate promising results in completing missing tags and predicting new tags. However, very limited prior research addresses the efficiency problem, while efficiency is a practical and critical issue in many large scale real world applications.

2.2 Learning Hashing Codes

Extensive research on learning hashing codes for fast similarity search [5, 22] have been proposed in recent years. Locality Sensitive Hashing (LSH) [5] utilizes random linear projections to map data examples from a high- dimensional Euclidean space to a low-dimensional one. LSH has been extended to Kernelized Locality Sensitive Hashing (KLSH) [12] by exploiting kernel similarity for better retrieval efficacy. The work in [16] uses stacked Restricted Boltzman Machine (RBM) to generate compact binary hashing codes for fast similarity search of documents. The PCA Hashing (PCAH) [15] method projects each example to the top principal components of the training set, and then binarizes the coefficients by setting a bit to 1 when its value is larger than the median value seen for the training set, and -1 otherwise.

Recently, Spectral Hashing (SH) [22] is proposed to learn compact hashing codes that preserve the similarity between data examples by balancing the hashing codes. Self Taught Hashing (STH) [24] combines an unsupervised learning step with a supervised learning step to learn effective hashing codes. More recently, work [25] investigates a new signature generation algorithm for learning hashing codes in content reuse detection. Most recently, Wang et al. [21] propose a semi-supervised learning framework to achieve binary hashing codes by using tags and topic modeling. One piece of research work related to our approach is [26], which constructs hashing codes for collaborative filtering (CFCode). Collaborative filtering can be viewed as a special kind of tag completion by treating users as data examples and items as tags.

These hashing methods only consider the problem of obtaining binary codes for data examples. However, in tag completion and prediction, hashing codes for both data examples and tags are involved and thus should be consider simultaneously, which makes it difficult to apply these method directly in this research work.

3. LEARNING HASHING CODES FOR TAG COMPLETING AND PREDICTION

3.1 Problem Setting and Overview

We first introduce the problem of HashTCP. Assume there are total n training examples in the dataset, denoted as: $x_i, i \in \{1, 2, \ldots, n\}$, where x_i is the d-dimensional feature of the *i-th* example. There are total m possible tags denoted as: $t_j, j \in \{1, 2, \ldots, m\}$. Denote the observed tag matrix as: $T \in \{0, 1\}^{n \times m}$, where a label $T_{ij} = 1$ means the *j-th* tag is assigned to the *i-th* example, and $T_{ij} = 0$ means a missing tag or the *i-th* example is not associated with the *j-th* tag. The main purpose of HashTCP is to obtain optimal binary hashing codes $y_i \in \{-1, 1\}^{C \times 1}, i \in \{1, 2, \ldots, n\}$ for the training examples and $z_j \in \{-1, 1\}^{C \times 1}, j \in \{1, 2, \ldots, m\}$ for all possible tags, where C is the code length. We also want to learn a hashing function $f : \boldsymbol{R}^d \rightarrow \{-1, 1\}^C$, which maps each example x_i to its hashing code y_i (i.e, $y_i = f(x_i)$).

3.2 Problem Formulation

The goal of tag completion and prediction is to complete missing tags for the training examples, and to predict tags for new query examples. Two main ingredients of constructing the compact hashing codes are: (1) A tag t_j should be assigned to a data example x_i if their corresponding hashing codes z_j and y_i are similar. (2) Similar data examples x_i and x_j should also have similar codes y_i and y_j. A natural way of defining the similarity between two hashing codes is to measure their normalized Hamming distance as follows:

$$s(y_i, z_j) = 1 - \frac{1}{C} dist_{Ham}(y_i, z_j) \tag{1}$$

where $dist_{Ham}$ is the Hamming distance between two binary hashing codes, which is given by the number of bits that are different between them. It can be seen from Eqn.1 that the smaller the Hamming distance is, the more similar their hashing codes become. Note that the similarity between two hashing codes is a real value in the range of 0 to 1.

The first key problem in designing hashing codes is to ensure the consistency between the observed tags and the constructed hashing codes. Specifically, we propose to minimize the squared loss of the observed tags and the similarity estimated from the hashing codes, which is a commonly used loss function in many machine learning applications.

$$\sum_{i=1}^{n} \sum_{j=1}^{m} (T_{ij} - s(y_i, z_j))^2 \tag{2}$$

As discussed before, $T_{ij} = 0$ can be interpreted in two ways that tag T_{ij} is missing or the *i-th* data example is not related to the *j-th* tag, which indicates that $T_{ij} = 1$ contains more useful information than a tag with value 0. Therefore, an importance matrix $I \in \boldsymbol{R}^{n \times m}$ is introduced to denote the confidence of how we trust tag information in tag matrix T. We set I_{ij} to a higher value a when $T_{ij} = 1$ and I_{ij} to a lower value b if $T_{ij} = 0$.[1]. Then the square loss term becomes:

$$\sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - s(y_i, z_j))^2 = I_{ij}(T_{ij} - \frac{1}{2} - \frac{y_i^T z_j}{2C})^2 \quad (3)$$

The other key problem in learning hashing codes is similarity preserving, which indicates that similar examples should be mapped to similar hashing codes within a short Hamming distance. To measure the similarity between data examples represented by the binary hashing codes, one natural way is to minimize the weighted average Hamming distance as follow:

$$\sum_{i,j=1}^{n} S_{ij}\ dist_{Ham}(y_i, y_j) = \frac{1}{2}\sum_{i,j=1}^{n} S_{ij}(C - y_i^T y_j) \quad (4)$$

Here, S is the similarity matrix which is calculated based on the features of the data examples. To meet the similarity preservation criterion, we seek to minimize this quantity, because it incurs a heavy penalty if two similar examples are mapped far away. There are many different ways of defining the similarity matrix S. In SH [22], the authors used the global similarity structure of all data pairs, while in [21] and [24], the local similarity structure, i.e., k-nearest-neighborhood, is used. In this paper, we adopt the local similarity.

The entire objective function of the proposed HashTCP approach consists of two components: the square loss of tag consistency term in Eqn.3 and the similarity preservation term given in Eqn.4 as follows:

$$\begin{aligned} &\min_{y,z} \sum_{i=1}^{n}\sum_{j=1}^{m} I_{ij}(T_{ij} - \frac{1}{2} - \frac{y_i^T z_j}{2C})^2 + \frac{\alpha}{2}\sum_{i,j=1}^{n} S_{ij}(C - y_i^T y_j) \\ &s.t. \qquad y_i, z_j \in \{-1, 1\}^{C\times 1},\ \sum_{i=1}^{n} y_i = 0\ \ \sum_{j=1}^{m} z_j = 0 \end{aligned} \quad (5)$$

where α is trade-off parameter between the two terms. The constraints $\sum_{i=1}^{n} y_i = 0$ and $\sum_{j=1}^{m} z_j = 0$ are the bit balance constraints, which require each bit of the hashing codes to have equal chance to be 1 or -1. The balance constraints are equivalent to maximizing the entropy of each bit of the binary hashing codes to ensure each bit carrying as much information as possible.

3.3 Optimization Algorithm

3.3.1 Relaxation

Directly minimizing the objective function in Eqn.5 is intractable since it is a constrained discrete optimization problem. Moreover, the balance constraints also make this problem NP-hard to solve [22]. Therefore, we propose to relax the balance constraints into soft penalty terms and then relaxing the space of solution to $[-1, 1]^{C\times 1}$. However, even after the relaxation, the objective function is still non-convex with respect to $\tilde{y}$ and $\tilde{z}$ jointly, which makes it difficult to optimize. Fortunately, this relaxed problem is differentiable with respect to either one of the two sets of parameters when the other one is fixed, and therefore we propose to solve the problem by coordinate descent method similar to [19, 20]. In particular, we alternatively update $\tilde{y}$ and $\tilde{z}$ while fixing the other set of parameters by doing the following two steps until convergence.

Step 1: Fix $\tilde{y}$, optimize $\tilde{z}$. By taking the partial derivative of Eqn.5 with respect to z_j, we can obtain the gradient below:

$$\partial\frac{Eqn.5}{\tilde{z_j}} = -\frac{1}{C}\sum_{i}\ I_{ij}(T_{ij} - \frac{1}{2} - \frac{\tilde{y_i}^T\tilde{z_j}}{2C})\tilde{y_i} + 2\beta\sum_{j'} \tilde{z_{j'}} \quad (6)$$

With this obtained gradient, LBFGS method [27] is applied for solving this optimization problem to obtain optimal $\tilde{z}$.

Step 2: Fix $\tilde{z}$, optimize $\tilde{y}$. By taking the partial derivative of Eqn.5 with respect to y_i, we can obtain the gradient as:

$$\begin{aligned} \partial\frac{Eqn.5}{\tilde{y_i}} = &-\frac{1}{C}\sum_{j}\ I_{ij}(T_{ij} - \frac{1}{2} - \frac{\tilde{y_i}^T\tilde{z_j}}{2C})\tilde{z_j} \\ &-\frac{\alpha}{2}\sum_{j} S_{ij}\tilde{y_j} + 2\beta\sum_{i'}\tilde{y_{i'}} \end{aligned} \quad (7)$$

Similar to step 1, we use LBFGS method with this gradient to solve for the optimal $\tilde{y}$.

3.3.2 Binarization

After obtaining the optimal real value solution $\tilde{y}$ and $\tilde{z}$ for the relax problem, we need to binarize them to obtain binary codes y and z. In this work, we propose a novel binarization method that improves the quantization error through an orthogonal transformation by making use of the structure of the relaxed solution as follows:

$$\begin{aligned} &\min_{y,z,Q} \sum_{i}\|y_i - Q\tilde{y_i}\|^2 + \sum_{j}\|z_j - Q\tilde{z_j}\|^2 \\ &s.t. \quad y_i, z_j \in \{-1, 1\}^{C\times 1}, \quad Q^T Q = I \end{aligned} \quad (8)$$

The above optimization problem can be solved by minimizing Eqn.8 with respect to y, z and Q alternatively.

Fix Q, update y and z. The close form solution can be expressed as:

$$y_i = sgn(Q\tilde{y_i}),\ z_j = sgn(Q\tilde{z_j}) \quad (9)$$

Fix y and z, update Q. The objective function becomes:

$$\min_{Q^T Q = I} \sum_{i}\|y_i - Q\tilde{y_i}\|^2 + \sum_{j}\|z_j - Q\tilde{z_j}\|^2 \quad (10)$$

In this case, the objective function is essentially a variant of classic Orthogonal Procrustes problem [17], which can be solved efficiently by singular value decomposition (we refer to [17] for details).

3.3.3 Hashing Function

In order to deal with the out-of-example problem in tag prediction, we propose to learn linear hashing function to map the data examples into binary hashing codes as:

$$y_i = f(x_i) = sgn(Hx_i) \quad (11)$$

[1]In our experiments, we set the importance parameters a=1 and b=0.01 consistently throughout all experiments.

*Flickr*1*m*	AP@10					AP@15					AP@20				
training tags	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
HashTCP	65.4	**68.9**	71.1	**76.7**	**80.4**	65.1	**67.8**	70.2	**74.3**	**76.7**	65.2	66.3	**70.4**	**73.6**	**75.3**
TMC[23]	**66.3**	68.6	**71.7**	76.2	79.5	**65.7**	67.7	**71.1**	73.5	75.9	**65.4**	**66.9**	69.3	72.1	74.7
LM3L[10]	60.4	65.8	68.3	71.6	74.7	60.3	65.1	66.9	69.6	72.4	58.5	62.0	65.8	68.7	70.8
CFCode[26]	62.9	64.1	66.8	71.7	73.4	60.9	64.3	65.4	69.3	71.6	57.2	61.8	62.7	66.4	70.1
STH[24]	55.2	57.5	59.7	61.1	64.6	54.0	55.8	57.2	59.8	61.4	53.3	55.2	56.3	57.8	60.2
SH[22]	53.7	55.3	57.5	58.4	60.7	53.1	54.6	56.2	56.9	59.8	52.4	53.8	55.1	55.6	57.5
NUS-WIDE	AP@10					AP@15					AP@20				
training tags	2	4	6	8	10	2	4	6	8	10	2	4	6	8	10
HashTCP	51.1	56.2	**67.4**	**74.7**	**80.5**	49.6	55.7	**64.6**	**73.0**	**78.4**	48.4	54.2	**61.8**	**70.2**	75.1
TMC[23]	**51.7**	**56.5**	66.4	72.5	76.7	**50.4**	**55.8**	64.2	69.9	75.0	**49.2**	**54.6**	59.4	67.5	**76.7**
LM3L[10]	48.3	53.1	61.4	72.0	73.6	47.0	54.7	61.4	68.8	71.3	46.6	51.7	58.4	62.9	67.7
CFCode[26]	47.6	53.4	59.1	70.6	74.0	48.3	50.9	56.6	62.1	68.6	47.2	52.0	58.1	60.5	64.8
STH[24]	44.8	49.5	54.7	58.4	62.2	43.6	47.2	51.7	56.0	61.5	42.3	46.1	49.6	54.8	59.0
SH[22]	43.2	47.0	52.9	56.8	58.3	41.5	45.9	49.8	53.4	58.6	40.7	43.8	47.2	51.5	56.1

Table 1: Performance of tag completion with varying number of training tags on two image datasets. The length of hashing code is fix to 32 for all hashing methods.

where H is a $C \times d$ parameter matrix representing the hashing function. Then the optimal hashing function can be obtained by:

$$\min_{H} \sum_{i} (\tilde{y}_i - Hx_i)^2 + \lambda \|H\|_F^2 \tag{12}$$

here λ is a weight parameter for the regularization term to avoid overfitting.

4. EXPERIMENTS

4.1 Datasets

We conduct our experiments on four datasets, including two image datasets and two text collections as follows:

*Flickr*1*m* [11] is collected from Flicker images containing 1 million image examples associated with more than $7k$ unique tags. A subset of $250k$ image examples with the most common $2k$ tags is used in our experiment by filtering out those images with less than 10 tags. 512-dimensional GIST descriptors are used as image features. We randomly choose $240k$ image examples as training set and $10k$ for query testing.

NUS-WIDE [4] is created by NUS lab containing $270k$ images associated with $5k$ possible tags. We use a subset of $110k$ image examples with $2k$ tags in our experiments. 500-dimensional visual features are extracted using a bag-of-visual-word model with local SIFT descriptor. We randomly partition this dataset into two parts, $10k$ for query testing and around $100k$ for training.

*ReutersV*1 (Reuters-Volume I) contains over $800k$ manually categorized newswire stories [13]. There are in total 126 different tags associated with this dataset. A subset of about $130k$ documents of *ReutersV*1 is used in our experiment by discarding those documents with less than 3 tags. $120k$ text documents are randomly selected as the training data, while the remaining $10k$ documents are used as testing queries.

Reuters (Reuters21578)[2] contains 21578 documents, and 135 tags. In our experiments, documents with less than 3 tags are removed. The remaining 13713 documents are randomly partitioned into training set with 12713 documents and 1000 test queries.

[2]http://daviddlewis.com/resources/textcollections/reuters21578/.

4.2 Comparison Methods

The proposed HashTCP approach is compared with five state-of-the-art methods, including two non-hashing methods TMC [23] and LM3L [10], and three hashing methods CFCode [26], STH [24] and SH [22].

Tag Matrix Completion (TMC) [23] directly completes the tag matrix by exploiting the tag correlation and data examples similarity to ensure the consistency between the observed tags and the estimated tags. Multi-Label Classification (LM3L) [10] proposes a large scale multi-label classification algorithm to overcome the training bias by incorporating correlation prior in a max-margin framework.

Binary Codes for Collaborative Filtering (CFCode) [26] constructs binary codes for collaborative filtering to recommend items to users. Self-Taught Hashing (STH) [24] combines an unsupervised learning step with a supervised learning step to learn effective hashing codes. Spectral Hashing (SH) [22] is proposed to learn compact hashing codes that preserve the similarity between data examples by balancing the hashing codes.

4.3 Evaluation on Tag Completion

In the first set of experiments, we evaluate the performance of different algorithms by varying the number of training tags. In particular, for image datasets, we vary the number of training tags for each image from $\{2, 4, 6, 8, 10\}$. For document datasets, since the tags associated with each document is relatively small, we vary the number of training tags for each document from $\{1, 2, 3\}$. We then rank the tags based on their relevance scores in the completed tag matrix, and return the top ranked tags. We use the average precision (AP) as the evaluation metric. Tables 1 summarizes the results for different methods on two image datasets. Note that for all hashing methods in this set of experiments, we fix the length of hashing codes to be 32. It is not surprising to see that the performance of all methods improve with the increasing number of training tags. From these comparison

	Flickr1m		*NUS-WIDE*		*ReutersV1*		*Reuters*	
method	training	tag completion	training	tag completion	training	tag completion	training	tag completion
HashTCP	352	1.23	126	0.38	63.75	0.24	16.93	0.05
TMC[23]	1537	N/A	528	N/A	234	N/A	56.34	N/A
LM3L[10]	489	23.52	154	7.86	72.87	4.91	27.48	1.27
CFCode[26]	337	1.31	108	0.39	58.49	0.24	15.50	0.05
STH[24]	214	1.23	81.83	0.37	45.92	0.25	12.47	0.05
SH[22]	198	1.22	79.44	0.38	37.69	0.26	11.56	0.05

Table 2: Training time and tag completion time (sec) for different methods on four datasets. The length of hashing code is fix to 32 for all hashing methods.

results, we can also see that HashTCP achieves the same or even better accuracy to the non-hashing methods (i.e., TMC and LM3L) and substantially outperforms all other hashing methods (i.e., CFCode, STH and SH). Similar results have been observed on the document datasets. Our hypothesis is that the compared hashing methods focus on encoding the consistency of the hashing codes to the observed tags without considering the relationship among the data examples, which tend to overfit. On the other hand, the proposed HashTCP constructs binary hashing codes by simultaneously ensuring the learned codes to be consistent with observed tags and preserving the similarity between data examples, which indicates that HashTCP generates more effective codes and completes the missing tags accurately.

In the second set of experiments, we evaluate the efficiency of different methods on four datasets. The training time and tag completion time are reported in Table 2. We also fix the hashing bits to be 32 for all hashing methods. For the TMC method, we only report the training time since the tag completion procedure is embedded in its training procedure. From the reported results, it is clear that tag completion process of hashing methods is 20 to 25 times faster than traditional multi-label learning method LM3L. This is because hashing methods use binary codes to calculate the tag relevance scores, which only involves efficient bit-wise operations XOR, while traditional methods need to deal with real value vectors to compute the tag scores. We also observe that the training time of our method is comparable with other hashing methods and is much faster than TMC since the learning algorithm of TMC is quite involved with multiple terms. We will discuss more in next section that our method can achieve sub-linear time for tag prediction using a fixed small radius Hamming ball.

4.4 Evaluation on Tag Prediction

In the first set of experiments, we evaluate the effectiveness of different algorithms on four datasets. We use the full observed tags as the training tags and predict tags on the test examples. CFCode is not compared here since it cannot be directly used to predict tags for unseen examples. For TMC, we use the same experiment setting in their work for tag prediction. We use average precision (AP) and average recall (AR) as the evaluation metrics. The performance results are given in Fig.1. We can see from this figure that these performance results for tag prediction are consistent with the results in Table 1 for tag completion. From these comparison results, we find that the precision usually declines with the increasing of the number of returned tags,

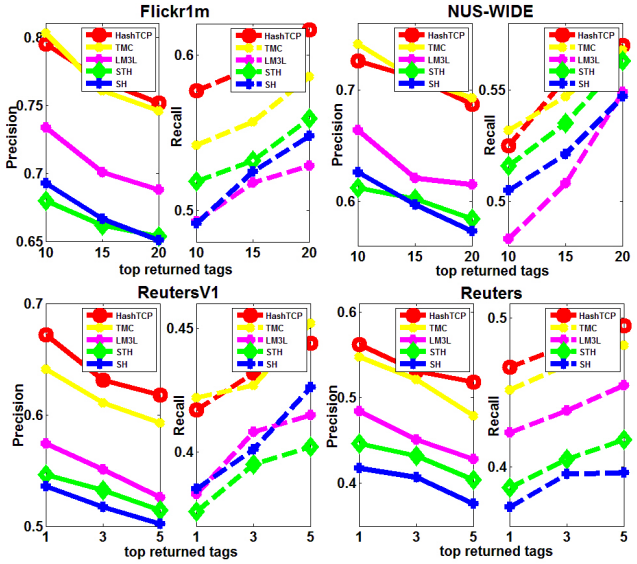

Figure 1: Performance of tag prediction for different methods on four datasets. The length of hashing code is fixed to be 32 for all hashing methods.

while the recall usually improves. This is called precision-recall tradeoff which is also observed in [23].

In the second set of experiments, we evaluate the efficiency of two prediction metrics *Hamming Ranking* and *Hash Lookup* on four datasets. *Hamming Ranking*, which we use in all our previous experiments, ranks all the tags according to their Hamming distance from the query example and the top k are returned as the desired tags. *Hash Lookup* returns all the tags within a small Hamming radius r of the query example. Note that *Hash Lookup* is not applicable for non-hashing methods. The comparison results of both two metrics are reported in Table 3. We choose top 10 and 5 tags on image and document datasets respectively for *Hamming Ranking*. Hamming radius 2 is used for *Hash Lookup*. From this table we can see that the results of *Hamming Ranking* is consistent with the results in Table 2 for tag completion. For *Hash Lookup*, it takes even less time than *Hamming Ranking* in two image datasets which is about 120 to 130 times faster compared with non-hashing methods. This is because *Hash Lookup* returns tags within a small Hamming radius of the query code, which can be performed in sub-linear time [16]. Therefore, tag prediction using *Hash Lookup* can be conducted by only going through a small

	*Flickr*1*m*		*NUS-WIDE*	
	top 10	radius 2	top 10	radius 2
HashTCP	0.008	0.002	0.008	0.001
TMC[23]	1.26	N/A	1.31	N/A
LM3L[10]	0.24	N/A	0.22	N/A
STH[24]	0.009	0.002	0.008	0.002
SH[22]	0.027	0.009	0.026	0.009
	*ReutersV*1		*Reuters*	
	top 5	radius 2	top 5	radius 2
HashTCP	0.0005	0.0004	0.0004	0.0004
TMC[23]	0.63	N/A	0.71	N/A
LM3L[10]	0.0122	N/A	0.0107	N/A
STH[24]	0.0005	0.0005	0.0004	0.0004
SH[22]	0.0013	0.0011	0.0014	0.0012

Table 3: Tag prediction time per query (sec) for different methods on four datasets. The length of hashing code is fix to 32 for all hashing methods.

fraction of all possible tags which significantly improves the efficiency. Furthermore, we conduct an additional experiment to evaluate the effectiveness of *Hash Lookup* for different hashing methods. The results shows that the proposed HashTCP substantially outperforms other hashing methods in *Hash Lookup* setting, which is consistent with our expectation.

5. CONCLUSION

This paper proposes a novel efficient approach for tag completion and prediction by designing compact binary hashing codes for both data examples and tags. In particular, each data example is represented by a C-bit binary code and each tag is also represented using a C-bit binary code. We then formulate the problem of learning binary hashing codes as a discrete optimization problem by simultaneously ensuring the observed tags to be consistent with the constructed hashing codes and preserving the similarities between data examples. Extensive experiments on four datasets, including two image datasets and two document collections, demonstrate that the proposed approach can achieve similar or even better accuracy with state-of-the-art methods while using much less time.

6. ACKNOWLEDGMENTS

This work is partially supported by NSF research grants IIS-0746830, CNS-1012208 and IIS-1017837. This work is also partially supported by the Center for Science of Information (CSoI), an NSF Science and Technology Center, under grant agreement CCF-0939370.

7. REFERENCES

[1] R. S. Cabral, F. D. la Torre, J. P. Costeira, and A. Bernardino. Matrix completion for multi-label image classification. In *NIPS*, pages 190–198, 2011.

[2] J.-F. Cai, E. J. Candès, and Z. Shen. A singular value thresholding algorithm for matrix completion. *SIAM Journal on Optimization*, 20(4):1956–1982, 2010.

[3] G. Chen, J. Zhang, F. Wang, C. Zhang, and Y. Gao. Efficient multi-label classification with hypergraph regularization. In *CVPR*, pages 1658–1665, 2009.

[4] T.-S. Chua, J. Tang, R. Hong, H. Li, Z. Luo, and Y. Zheng. Nus-wide: a real-world web image database from national university of singapore. In *CIVR*, 2009.

[5] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *Symposium on Computational Geometry*, pages 253–262, 2004.

[6] C. Desai, D. Ramanan, and C. Fowlkes. Discriminative models for multi-class object layout. In *ICCV*, pages 229–236, 2009.

[7] J. Gemmell, T. Schimoler, B. Mobasher, and R. D. Burke. Hybrid tag recommendation for social annotation systems. In *CIKM*, pages 829–838, 2010.

[8] M. Guillaumin, T. Mensink, J. J. Verbeek, and C. Schmid. Tagprop: Discriminative metric learning in nearest neighbor models for image auto-annotation. In *ICCV*, pages 309–316, 2009.

[9] Y. Guo and S. Gu. Multi-label classification using conditional dependency networks. In *IJCAI*, pages 1300–1305, 2011.

[10] B. Hariharan, L. Zelnik-Manor, S. V. N. Vishwanathan, and M. Varma. Large scale max-margin multi-label classification with priors. In *ICML*, pages 423–430, 2010.

[11] M. J. Huiskes, B. Thomee, and M. S. Lew. New trends and ideas in visual concept detection: the mir flickr retrieval evaluation initiative. In *Multimedia Information Retrieval*, pages 527–536, 2010.

[12] B. Kulis and K. Grauman. Kernelized locality-sensitive hashing for scalable image search. In *ICCV*, pages 2130–2137, 2009.

[13] D. D. Lewis, Y. Yang, T. G. Rose, and F. Li. Rcv1: A new benchmark collection for text categorization research. *Journal of Machine Learning Research*, 5:361–397, 2004.

[14] X. Li, C. G. M. Snoek, and M. Worring. Learning social tag relevance by neighbor voting. *IEEE Transactions on Multimedia*, 11(7):1310–1322, 2009.

[15] R.-S. Lin, D. A. Ross, and J. Yagnik. Spec hashing: Similarity preserving algorithm for entropy-based coding. In *CVPR*, pages 848–854, 2010.

[16] R. Salakhutdinov and G. E. Hinton. Semantic hashing. *Int. J. Approx. Reasoning*, 50(7):969–978, 2009.

[17] P. Schonemann. A generalized solution of the orthogonal procrustes problem. *Psychometrika*, 31(1):1–10, 1966.

[18] Y. Song, B. Qiu, and U. Farooq. Hierarchical tag visualization and application for tag recommendations. In *CIKM*, pages 1331–1340, 2011.

[19] Q. Wang, L. Si, and D. Zhang. A discriminative data-dependent mixture-model approach for multiple instance learning in image classification. In *ECCV (4)*, pages 660–673, 2012.

[20] Q. Wang, L. Tao, and H. Di. A globally optimal approach for 3d elastic motion estimation from stereo sequences. In *ECCV (4)*, pages 525–538, 2010.

[21] Q. Wang, D. Zhang, and L. Si. Semantic hashing using tags and topic modeling. In *SIGIR*, pages 213–222, 2013.

[22] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. In *NIPS*, pages 1753–1760, 2008.

[23] L. Wu, R. Jin, and A. K. Jain. Tag completion for image retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(3):716–727, 2013.

[24] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *SIGIR*, pages 18–25, 2010.

[25] Q. Zhang, Y. Wu, Z. Ding, and X. Huang. Learning hash codes for efficient content reuse detection. In *SIGIR*, pages 405–414, 2012.

[26] K. Zhou and H. Zha. Learning binary codes for collaborative filtering. In *KDD*, pages 498–506, 2012.

[27] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal. Algorithm 778: L-bfgs-b: Fortran subroutines for large-scale bound-constrained optimization. *ACM Trans. Math. Softw.*, 23(4):550–560, 1997.