Semantic Hashing using Tags and Topic Modeling

Qifan Wang
Computer Science Department
Purdue University
West Lafayette, IN 47907, US
wang868@purdue.edu

Dan Zhang
Facebook Incorporation
Menlo Park, CA 94025, US
danzhang@fb.com

Luo Si
Computer Science Department
Purdue University
West Lafayette, IN 47907, US
lsi@purdue.edu

ABSTRACT

It is an important research problem to design efficient and effective solutions for large scale similarity search. One popular strategy is to represent data examples as compact binary codes through semantic hashing, which has produced promising results with fast search speed and low storage cost. Many existing semantic hashing methods generate binary codes for documents by modeling document relationships based on similarity in a keyword feature space. Two major limitations in existing methods are: (1) Tag information is often associated with documents in many real world applications, but has not been fully exploited yet; (2) The similarity in keyword feature space does not fully reflect semantic relationships that go beyond keyword matching.

This paper proposes a novel hashing approach, Semantic Hashing using Tags and Topic Modeling (SHTTM), to incorporate both the tag information and the similarity information from probabilistic topic modeling. In particular, a unified framework is designed for ensuring hashing codes to be consistent with tag information by a formal latent factor model and preserving the document topic/semantic similarity that goes beyond keyword matching. An iterative coordinate descent procedure is proposed for learning the optimal hashing codes. An extensive set of empirical studies on four different datasets has been conducted to demonstrate the advantages of the proposed SHTTM approach against several other state-of-the-art semantic hashing techniques. Furthermore, experimental results indicate that the modeling of tag information and utilizing topic modeling are beneficial for improving the effectiveness of hashing separately, while the combination of these two techniques in the unified framework obtains even better results.

Categories and Subject Descriptors

H.3.1 [**Information Storage and Retrieval**]: Content Analysis and Indexing; H.3.3 [**Information Storage and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning

General Terms

Algorithms, Performance, Experimentation

Keywords

Hashing, Tags, Topic Modeling

1. INTRODUCTION

Similarity search identifies similar information objects given a query object, which has many information retrieval applications such as similar document detection, content-based image retrieval and collaborative filtering. Due to the explosive growth of the internet, a huge amount of data such as texts, images and videos has been generated, which indicates that nearest neighbor methods for similarity search are becoming more reliable. Therefore, it is important to design effective and efficient nearest neighbor methods for similarity search with large scale data.

Two major challenges have to be addressed for using similarity search in large scale datasets such as storing the data efficiently and retrieving the large scale data in an effective and efficient manner. Traditional text similarity search methods in the original keyword vector space are difficult to be used for large datasets, since these methods utilize the content vectors of the documents in a high-dimensional space and are associated with high cost of float/integer computation.

Semantic hashing (e.g., [9, 22, 34]) has been proposed as a promising technique for addressing these two challenges, which designs compact binary code in a low-dimensional space for each document so that similar documents are mapped to similar binary codes. Query documents can also be efficiently transformed into hashing codes so that similarity search can be conducted. The retrieval process of similarity search can be simply conducted by calculating the Hamming distances between the hashing codes of available documents and a query and selecting documents within small Hamming distance. Therefore, this method addresses the two major challenges of large scale similarity search in the following ways: (1) The encoded data is highly compressed within a low-dimensional binary space, and thus can often be dealt with in main memory and stored efficiently; (2) The retrieval process is very efficient, since the distance between two codes is simply the number of bits that they differ.

Existing semantic hashing approaches generate promising results and are successful in addressing the two challenges of storage and retrieval efficiency. However, two major issues are not addressed in the existing methods: (1) Tag information is not fully utilized in previous methods. Most existing methods only deal with the contents of documents without utilizing the information contained in tags. Actually, in many real-world applications, documents are often associated with multiple tags, which provide useful knowledge in learning effective hashing codes. For instance, in some big corporations, their employees often assign tags to some webpages in the intranet, such that they can find these webpages again more easily through these tags. Another example is microblog, usually each tweet is associated with multiple tags such as 'stock', 'business', 'sports', etc.; (2) Document similarity in the original keyword feature space is used as guidance for generating hashing codes in previous methods, which may not fully reflect the semantic relationship. For example, two documents in the same topic may have low document content similarity in keyword space due to the vocabulary gap, although their semantic similarity can be high.

Based on the above observations, this paper proposes a novel hashing approach, Semantic Hashing using Tags and Topic Modeling (SHTTM). The SHTTM approach integrates available tag information and document similarity in semantic topics for generating effective hashing codes. There are three main challenges for designing this method: (1) How to incorporate the tag information? (2) How to preserve the similarity between documents based on semantic topics? (3) How to obtain the hashing code for a query document as no tag information is available for the query document?

A unified framework is proposed to address the above challenges, which ensures the consistency of hashing codes with tag information by a latent factor model and preserves document semantic similarity based on topic modeling. An iterative optimization procedure is proposed for learning the optimal hashing codes. An extensive set of experiments on four real world datasets has been conducted to demonstrate the advantages of the proposed method over several state-of-the-art methods. More specifically, a set of experiments clearly demonstrates the benefits of utilizing tag information and topic modeling separately, while the combination of these two methods in the unified framework generates the best results. To our best knowledge, this is the first piece of research work on semantic hashing that integrates both tag information and semantic topic modeling, which generates more effective hashing codes than several other state-of-the-art methods.

2. RELATED WORK

Efficiency is a crucial issue for large scale information retrieval applications with a huge amount of text documents. When there is only a low-dimensional feature space, similarity search can be carried out by some space partitioning index structures, such as TF-IDF methods [23, 24], KD-tree, or data partitioning index structures, like R-tree [5]. Several types of structures and operations of inverted indexing are also proposed [6, 26, 40] for traditional ad-hoc text search with relatively short user queries. However, traditional similarity search may fail to work efficiently within a high-dimensional vector space [33], which is often the case for many real world information retrieval applications.

Semantic hashing [22] is proposed to address the similarity search problem within a high-dimensional feature space. In particular, semantic hashing methods try to represent each document by using a small fixed number of binary bits so that the queries can be answered in a short time [27]. The hashing based fast similarity search can be viewed as a strategy to transform documents from a high-dimensional space into a low-dimensional binary space, and at the same time preserve the semantic similarity between documents as much as possible. Hashing methods generate binary codes for efficient search, which is different from traditional dimensionality reduction methods such as Principal Component Analysis (PCA) and Latent Semantic Indexing (LSI) [8, 12].

Locality-Sensitive Hashing (LSH) [2, 7] is one of the most popularly used hashing methods. It simply utilizes random linear projections to map documents from a high-dimensional Euclidean space to a low-dimensional one. It has already been shown that the Hamming distance between different documents will asymptotically approach their Euclidean distance in the original feature space with the increase of the hashing bits. LSH has been extended to Kernelized Locality-Sensitive Hashing (KLSH) [16] by exploiting kernel similarity for better retrieval efficacy. Recently, the work in [35] further extends the KLSH to the scheme of Boosting Multi-Kernel Locality-Sensitive Hashing (BMKLSH) that improves the retrieval performance of KLSH by making use of multiple kernels.

Several machine learning approaches have been proposed to solve the hashing problem. For example, the PCA Hashing [19] 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. The work in [22] uses stacked Restricted Boltzman Machine (RBM) [10, 11] to generate compact binary hashing codes, which can be viewed as binarized LSI. Recently, Spectral Hashing (SH) [34] is proposed to learn compact binary codes that preserve the similarity between documents by forcing the balanced and uncorrelated constraints into the learned codes, which be viewed as an extension of spectral clustering [37]. A graph-based hashing method has been proposed in work [21] to automatically discover the neighborhood structure inherent in the data to learn appropriate compact codes. More recently, the work [38] proposes a Composite Hashing with Multiple Information Sources (CHMIS) method to integrate information from different sources. In another recent work [15], an isotropic hashing (IsoHash) method is proposed to learn projection functions of individual hashing codes with equal variances.

In the following two sections, we mainly discuss two state-of-the-art hashing methods that are most related to the proposed research as Self-Taught Hashing [39] and Semi-Supervised Hashing [31].

2.1 Self-Taught Hashing(STH)

Self Taught Hashing (STH) [39] generally provides more effective hashing solutions than LSH [7] and SH [34]. STH combines an unsupervised learning step with a supervised learning step to learn hashing codes.

In the unsupervised learning step, STH constructs a similarity graph using a fix number of nearest neighbors for the given dataset, and then embeds all the documents into a k dimensional space through spectral analysis, and finally uses simple thresholding to obtain the binary hashing code for each document. This step can be formulated as follows:

$$\begin{aligned} &\min \sum_{i,j} \boldsymbol{S}_{ij} ||y_i - y_j||^2 \\ &s.t. \;\; y_i \in \{-1,1\}^k, \sum_i y_i = 0, \frac{1}{n}\sum_i y_i y_i^T = \boldsymbol{I} \end{aligned} \tag{1}$$

$\boldsymbol{S}_{ij}$ is the pairwise similarity between document i and document j, y_i is the hashing code for document i, k is the number of hashing bits. The objective function incurs a heavy penalty if two similar documents are mapped far away, which preserves the similarity between documents. The constraint $\sum_i y_i = 0$ requires each bit to be balanced and $\frac{1}{n}\sum_i y_i y_i^T = \boldsymbol{I}$ forces the hashing bits to be uncorrelated with each other.

In the supervised learning step, a set of k SVM classifiers are trained based on existing documents and their binary hashing codes learned from the previous step. Then, the k classifiers can be used to generate the hashing codes for the query documents as a classification problem. STH does not assume that data are uniformly distributed in a hyper-rectangle as requested by SH, which is often too restrictive for real world applications. STH often generates more effective hashing codes than SH.

However, there are two main limitations for STH. Firstly, STH does not utilize tag information, which is often available with documents in many real world applications. Secondly, the similarity matrix in STH is calculated in the original feature (keyword vector) space, which may not reflect the semantic similarity beyond simple keyword matching. Topic modeling (e.g., [4]) has been shown as an effective approach for capturing semantic meanings in text documents. In the proposed new research, we address both of the two problems of STH through integrating the tag information and topic modeling into one unified framework so that the semantic similarity between documents can be better preserved in the learned hashing codes.

2.2 Semi-Supervised Hashing(SSH)

The work in [31] proposes a Semi-Supervised Hashing (SSH) approach for incorporating the pairwise relationships between documents into the semantic hashing problem. More precisely, these pairwise similarity constraints are also called Must-Link and Cannot-Link, which could be partially generated from tags. For example, a Must-Link is created when two documents share a common tag and a Cannot-Link is created when two documents share no tag. Their basic motivation is that the hashing codes of the document pairs with Must-Link should be as close as possible, while the hashing codes of document pairs with Cannot-Link should be as different as possible. This motivation is then incorporated into the objective function for learning the hashing codes. A sequential projection method is utilized to solve the resulting optimization problem.

The SSH has shown promising results for improving hashing effectiveness by leveraging the pairwise information, but there are several limitations for SSH. Firstly, the SSH method only utilizes the pairwise similarity constraints as the summary of tag information, which is suboptimal with respect to the complete information in the tags. Secondly, the pairwise link information may not be accurately generated when tags are missing, incomplete or mismatched, which is often the case for many real world applications. Furthermore, SSH also directly works in the original keyword feature space for modeling content similarity of documents. These problems may potentially limit the performance of the hashing methods based on pairwise constraints. Different from SSH, the proposed method utilizes the complete tag information as well as semantic information from topic modeling for building more effective hashing codes.

3. ALGORITHM DESCRIPTION

3.1 Problem Setting

We first introduce the problem of SHTTM. Assume there are total n training documents in the dataset, denoted as: $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\} \in \boldsymbol{R}^{m \times n}$, where m is the dimensionality of the content feature. Denote their tags as: $\boldsymbol{T} = \{t_1, t_2, \ldots, t_n\} \in \{0,1\}^{l \times n}$, where l is the total number of possible tags associated with each document. A tag with label 1 means a document is associated with a certain tag/category, while a tag with label 0 means a missing tag or the document is not associated with that tag/category. The main purpose of SHTTM is to obtain optimal binary hashing codes $\boldsymbol{Y} = \{y_1, y_2, \ldots, y_n\} \in \{-1,1\}^{k \times n}$ for the training documents $\boldsymbol{X}$, and a hashing function $f : \boldsymbol{R}^m \rightarrow \{-1,1\}^k$, which maps each document to its hashing code with k bits (i.e, $y_j = f(x_j)$).

3.2 Approach Overview

The proposed SHTTM approach is a general learning framework that consists of two stages. In the first stage, the hashing codes are learned in a unified framework by simultaneously ensuring hashing codes to be consistent with tag information by a formal latent factor model and preserving the document topic/semantic similarity. In particular, the objective function of SHTTM is composed of two components: (1) Tag consistency component, which ensures that the hashing codes are consistent with tag information; (2) Similarity preservation component, which aims at preserving the topic/semantic similarity in the learned hashing codes. An iterative algorithm is then derived based on the objective function using a coordinate descent optimization procedure. In the second stage, the hashing function is learned with respect to the hashing codes for training documents.

The rest of this section first presents the two stages of the proposed SHTTM approach respectively, and then addresses the corresponding optimization problem. Finally, this section discusses connections and distinctions of the proposed approach with some related research work.

3.3 Tag Consistency

In many real world applications, documents are associated with tag information, which can provide useful knowledge in learning effective hashing codes. There are two main challenges for utilizing tags: (1) We have no knowledge about how the tags are related to the hashing codes. Therefore, we need to explore the correlation between them in order to bridge tags with hashing codes; (2) Tags may be missing, we need to deal with the situation of incomplete tags.

The first problem of exploring the correlation between tags and hashing codes can be addressed by matrix factorization with a latent factor model. A latent variable u_i for each tag t_i is first introduced, where u_i is a $k \times 1$ vector indicating the correlation between tags and hashing codes. Then a tag consistency component can be formulated as follows:

$$\sum_{i=1}^{l}\sum_{j=1}^{n} \|\boldsymbol{T}_{ij} - u_i^T y_j\|^2 + \alpha \sum_{i=1}^{l} \|u_i\|^2 \tag{2}$$

where $\boldsymbol{T}_{ij}$ is the binary label of the *i-th* tag on the *j-th* document. $u_i^T y_j$ can be viewed as a weighted sum that indicates how the *i-th* tag is related to the *j-th* document, and this weighted sum should be consistent with the observed label $\boldsymbol{T}_{ij}$ as much as possible. The second regularization component, $\sum_{i=1}^{l} \|u_i\|^2$, is introduced to avoid the overfitting issue (e.g., [28, 29]). α is a meta parameter that explores the trade-off between the tag consistence and regularization.

The second problem of dealing with missing tags can be addressed by introducing a confidence matrix $\boldsymbol{C} \in \boldsymbol{R}^{l\times k}$. If the value of $\boldsymbol{C}_{ij}$ is large, we trust the tag information $\boldsymbol{T}_{ij}$ more. As discussed before, $\boldsymbol{T}_{ij} = 0$ can be interpreted in two ways: tag i on the *j-th* document is either missing or not related. We will use a similar strategy as in [14] for a different application to set $\boldsymbol{C}_{ij}$ a higher value when $\boldsymbol{T}_{ij} = 1$ than $\boldsymbol{T}_{ij} = 0$ as follows,

$$\boldsymbol{C}_{ij} = \begin{cases} a, & if\ \boldsymbol{T}_{ij} = 1 \\ b, & if\ \boldsymbol{T}_{ij} = 0 \end{cases} \tag{3}$$

where a and b are parameters satisfying $a > b > 0$[1]. Then the whole component of tag consistency becomes:

$$\sum_{i=1}^{l}\sum_{j=1}^{n} \boldsymbol{C}_{ij}\|\boldsymbol{T}_{ij} - u_i^T y_j\|^2 + \alpha \sum_{i=1}^{l} \|u_i\|^2 \tag{4}$$

The above equation can be rewritten in a compact matrix form as:

$$\|\boldsymbol{C}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{U}^T\boldsymbol{Y})\|_F^2 + \alpha\|\boldsymbol{U}\|^2 \tag{5}$$

where $\boldsymbol{C}^{\frac{1}{2}}$ is the element-wise square root matrix of $\boldsymbol{C}$, and $\cdot$ is the element-wise matrix multiplication. $\|\|_F$ is the matrix Frobenius norm. By minimizing this component, the consistency between tags and the learned hashing codes can be ensured.

3.4 Topic Modeling and Similarity Preservation

One of the key problems in semantic hashing methods is similarity preserving, which indicates that semantically similar documents should be mapped to similar hashing codes within a short Hamming distance. A popular criterion of similarity preservation is to minimize the objective value in Eqn.1, where $\boldsymbol{S}$ is the document similarity matrix. This criterion incurs a heavy penalty if two similar documents are mapped far away in the dataset.

There are many different ways of defining the similarity matrix. In SH [34], the authors used the global similarity structure of all document pairs, while in STH [39] and CHMIS [38], the local similarity structure,*i.e.*, k nearest neighborhood, is used. However, all these methods compute the similarity matrix $\boldsymbol{S}$ within the original keyword feature space and thus may not reflect the document semantic similarity that goes beyond simple keyword matching.

To address the limitation of calculating semantic similarity in existing approaches, features from topic modeling are used to measure the semantic similarity between documents instead of features from the original keyword space. Topic modeling algorithms (e.g., [3, 36]) are used to discover a set of "topics" from a large collection of documents and provide an interpretable low-dimensional representation of the documents associated with the topics. Topic modeling has been widely used in many information retrieval applications such as document clustering and classification. Here we exploit the Latent Dirichlet Allocation (LDA) [4] approach of topic modeling to extract k latent topics from the document corpus. Each document x_j corresponds to a distribution θ_j over the topics where two semantically similar documents have similar topic distributions. In this way, document semantic similarity is preserved in the extracted topic distributions $\boldsymbol{\theta}$. Since we require the hashing codes to reflect the topic distributions, a document similarity preservation component can be naturally defined as follows:

$$\sum_{j=1}^{n} ||y_j - \theta_j||^2 = ||\boldsymbol{Y} - \boldsymbol{\theta}||^2 \tag{6}$$

By minimizing this component, the similarity between different documents is preserved in the learned hashing codes.

3.5 Overall Objective and Optimization Algorithm

The entire objective function of the proposed SHTTM approach integrates two components such as the tag consistency component in Eqn.5 and the semantic similarity preservation component given in Eqn.6 as follows:

$$\begin{aligned} &\min_{\boldsymbol{Y},\boldsymbol{U}} \|\boldsymbol{C}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{U}^T\boldsymbol{Y})\|_F^2 + \alpha\|\boldsymbol{U}\|^2 + \gamma\|\boldsymbol{Y} - \boldsymbol{\theta}\|^2 \\ &s.t. \qquad \boldsymbol{Y} \in \{-1, 1\}^{k\times n}, \ \ \boldsymbol{Y}\boldsymbol{1} = 0 \end{aligned} \tag{7}$$

where α and γ are trade-off meta parameters to balance the weight between the components. The constraint $\boldsymbol{Y}\boldsymbol{1} = 0$ requires each bit to appear 50% of the time (with equal probability as positive or negative). Note that one advantage of our method is that we do not need the bits-uncorrelated constraint $\boldsymbol{Y}\boldsymbol{Y}^T = \boldsymbol{I}$ while many previous methods do. This is because we assign each hashing bit a semantic meaning (a latent topic), which is learned from topic modeling and latent topics can be assumed to be highly uncorrelated. Therefore, by imposing the semantic similarity term between $\boldsymbol{Y}$ and $\boldsymbol{\theta}$, the uncorrelated property among $\boldsymbol{Y}$ is preserved.

Directly minimizing the objective function in Eqn.7 is intractable because of the discrete constraints. Therefore, we propose to relax this constraint and drop the constraint $\boldsymbol{Y}\boldsymbol{1} = 0$ first (we will discuss this constraint later). However, even after the relaxation, the objective function is still non-convex with respect to $\boldsymbol{Y}$ and $\boldsymbol{U}$ jointly, which makes it difficult to optimize. Fortunately, this relaxed problem is convex with respect to either one of the two sets of parameters when the other one is fixed, and therefore can be

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

```
Training procedure:
Input:
1. A set of $n$ training documents, $\boldsymbol{X} = \{x_1, x_2, \ldots, x_n\}$ associated with tags $\boldsymbol{T} = \{t_1, t_2, \ldots, t_n\}$.
2. Meta parameters: Tag consistency parameter $\alpha$, Similarity preservation parameter $\gamma$
and Hashing function parameter $\lambda$.
Output:
Hashing codes $\boldsymbol{Y}$ for the training documents, Hashing function matrix $\boldsymbol{W}$, Tag-Hashing code correlation
matrix $\boldsymbol{U}$ and Median vector $\boldsymbol{m}$.
Initialization:
1. Topic modeling: learning topic distribution of training documents $\boldsymbol{\theta} = LDA(\boldsymbol{X})$.
2. Construct confidence matrix $\boldsymbol{C}$ by Eqn.3.
3. Initialize $\boldsymbol{Y} = \boldsymbol{\theta}$.
4. Coordinate Descent Optimization: obtain the optimal $\boldsymbol{Y}$ and $\boldsymbol{U}$ by iteratively solving
Eqn.9 and Eqn.11 and updating $\boldsymbol{U}$ and $\boldsymbol{Y}$ respectively.
5. Optimizing Eqn.13 to obtain the hashing function matrix $\boldsymbol{W}$.
6. Get the median vector by $\boldsymbol{m} = median(\boldsymbol{Y})$.
7. Generating the hashing codes by thresholding $\boldsymbol{Y}$ to the median vector $\boldsymbol{m}$.
Predicting procedure:
Input:
A query document $q$, Hashing function matrix $\boldsymbol{W}$ and Median vector $\boldsymbol{m}$.
Output:
The binary hashing code $y_q$ for $q$.
1. Calculate the regression output by hashing function $f(q) = \boldsymbol{W}q$.
2. Obtain the hashing code $y_q$ by thresholding $f(q)$ to the median vector $\boldsymbol{m}$.
```

Table 1: The algorithm description of the complete SHTTM approach.

solved by coordinate descent optimization with guaranteed convergence similar to [32]. In particular, after initializing $\boldsymbol{Y}$, the optimization problem can be solved by doing the following two steps iteratively, until convergence.

Step 1: Fix $\boldsymbol{Y}$, optimize:

$$\min_{\boldsymbol{U}} \|\boldsymbol{C}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{U}^T\boldsymbol{Y})\|_F^2 + \alpha\|\boldsymbol{U}\|^2 \tag{8}$$

By taking the derivative of Eqn.8 with respect to u_i and setting it to 0, we can obtain the close form solution of this optimization below:

$$u_i = (\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{Y}^T + \alpha\boldsymbol{I})^{-1}\boldsymbol{Y}\boldsymbol{C}_i\boldsymbol{T}_i \tag{9}$$

where $\boldsymbol{C}_i$ is a $n \times n$ diagonal matrix with $\boldsymbol{C}_{ij}, j = 1, 2, \ldots, n$ as its diagonal elements and $\boldsymbol{T}_i = (\boldsymbol{T}_{ij}), j = 1, 2, \ldots, n$ is a $n \times 1$ label vector of *i-th* tag.

Step 2: Fix $\boldsymbol{U}$, optimize:

$$\min_{\boldsymbol{Y}} \|\boldsymbol{C}^{\frac{1}{2}} \cdot (\boldsymbol{T} - \boldsymbol{U}^T\boldsymbol{Y})\|_F^2 + \gamma\|\boldsymbol{Y} - \boldsymbol{\theta}\|^2 \tag{10}$$

By taking the derivative of Eqn.10 with respect to y_j and setting it to 0, we can obtain the optimal y_j:

$$y_j = (\boldsymbol{U}\boldsymbol{C}_j\boldsymbol{U}^T + \gamma\boldsymbol{I})^{-1}(\boldsymbol{U}\boldsymbol{C}_j\boldsymbol{T}_j + \gamma\boldsymbol{\theta}_j) \tag{11}$$

where $\boldsymbol{C}_j$ is a $l \times l$ diagonal matrix with $\boldsymbol{C}_{ij}, i = 1, 2, \ldots, l$ as its diagonal elements and $\boldsymbol{T}_j = (\boldsymbol{T}_{ij}), i = 1, 2, \ldots, l$ is a $l \times 1$ label vector of the *j-th* document. By solving Eqns.8 and 10 iteratively, optimal $\boldsymbol{Y}$ and $\boldsymbol{U}$ can be obtained.

3.6 Hashing Function

In the previous section, the optimal hashing codes $\boldsymbol{Y}$ are obtained after relaxing the binary constraint and the bit balance constraint $\boldsymbol{Y}\boldsymbol{1} = 0$. In this section, we will discuss how to obtain the hashing function that maps the data into binary hashing codes and how to binarize the optimal $\boldsymbol{Y}$ in order to satisfy the constraints.

In this paper, we utilize a linear hashing function to generate binary hashing code, which is consistent with previous hashing methods (e.g., [30, 31, 38]) as:

$$y_j = f(x_j) = \boldsymbol{W}x_j \tag{12}$$

where $\boldsymbol{W}$ is a $k \times m$ parameter matrix representing the hashing function. Then the optimal hashing function can be obtained by minimizing the following objective:

$$\begin{aligned} \boldsymbol{W}^{\star} &= \arg\min_{\boldsymbol{W}} \sum_{j=1}^{n} \|y_j - \boldsymbol{W}x_j\|^2 + \lambda\|\boldsymbol{W}\|_F^2 \\ &\Rightarrow \boldsymbol{W}^{\star} = \boldsymbol{Y}^T\boldsymbol{X}(\boldsymbol{X}^T\boldsymbol{X} + \lambda\boldsymbol{I})^{-1} \end{aligned} \tag{13}$$

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

The binary hashing codes for the training set can be obtained by thresholding $\boldsymbol{Y}$. Then, a natural question is how to pick these thresholds? In [19] and [37], the authors pointed out that a good semantic hashing should also maximize the entropy to ensure efficiency. Following the maximum entropy principle, a binary bit that gives balanced partitioning of the whole dataset always provides maximum information. Therefore, we set the threshold for binarizing the *p-th* bit to be the median of y^p. We denote the median of all bits by vector $\boldsymbol{m}$. Thus, if the *p-th* bit of document y_j is larger than $\boldsymbol{m}_p$, then y_j^p is set to +1, otherwise y_j^p is set to -1. In this way, the binary code achieves the best balance and the constraint $\boldsymbol{Y}\boldsymbol{1} = 0$ in Eqn.7 can also be satisfied. The hashing code of a query document q can be obtained by first computing the hashing function $f(q) = \boldsymbol{W}q$. Then, the

	ReutersV1					*Reuters*				
Methods	8 bits	16 bits	32 bits	64 bits	128 bits	8 bits	16 bits	32 bits	64 bits	128 bits
SHTTM	**0.559**	0.636	**0.712**	**0.781**	**0.815**	0.716	**0.741**	**0.755**	**0.758**	**0.753**
SSH [31]	0.531	**0.641**	0.664	0.722	0.727	**0.718**	0.722	0.724	0.731	0.739
STH [39]	0.507	0.568	0.643	0.646	0.694	0.703	0.715	0.731	0.740	0.734
SH [34]	0.514	0.556	0.617	0.631	0.658	0.641	0.694	0.711	0.703	0.716
PCAH [19]	0.431	0.557	0.601	0.638	0.641	0.652	0.687	0.693	0.646	0.631
LSH [7]	0.289	0.382	0.444	0.557	0.652	0.569	0.592	0.624	0.653	0.678

	20Newsgroups					*WebKB*				
Methods	8 bits	16 bits	32 bits	64 bits	128 bits	8 bits	16 bits	32 bits	64 bits	128 bits
SHTTM	**0.576**	**0.644**	**0.648**	**0.684**	**0.697**	**0.546**	**0.558**	**0.581**	**0.611**	**0.625**
SSH [31]	0.557	0.576	0.624	0.649	0.665	0.514	0.536	0.543	0.562	0.583
STH [39]	0.526	0.585	0.615	0.623	0.651	0.421	0.449	0.495	0.532	0.538
SH [34]	0.510	0.579	0.574	0.607	0.622	0.504	0.513	0.536	0.541	0.547
PCAH [19]	0.471	0.513	0.541	0.568	0.616	0.471	0.524	0.531	0.568	0.570
LSH [7]	0.389	0.431	0.474	0.557	0.646	0.339	0.377	0.389	0.387	0.401

Table 2: Precision of the top 100 retrieved documents on four datasets with different hashing bits.

corresponding binary hashing code can be obtained through thresholding the hashing function output, *i.e.*, set the j-th code for q to be +1, if $f(q)_j > \boldsymbol{m}_j$, and -1 otherwise.

The whole training procedure and predicting procedure of the proposed method is described in Table 1. For hashing methods, the training process is always conducted off-line. Therefore, our focus of efficiency is on the predicting process. This process of generating hashing code for a query for prediction only involves some dot products and comparisons between two binary vectors, which can be done in $O(m * k + k)$ time.

3.7 Discussion

The formulation of the proposed SHTTM approach is related to the traditional collaborative filtering methods (e.g., [14, 25, 13]) by treating each document as an item and the label of each tag as a rating from a user (i.e., tag) on this item (i.e., document). From this perspective, the proposed method is related to Collaborative Topic Modeling (CTM) [29], in which both the rating scores and the content of items are incorporated into one topic modeling framework. However, several major differences exist between the proposed work and CTM: (1). CTM is designed for collaborative filtering, while the proposed method investigates how tags and topic modeling can be integrated to improve the performance of hashing; (2). in the proposed research, one focus is on out-of-sample query examples, which do not exist in the training dataset. Therefore, the proposed research designs hashing function to convert query examples to their corresponding hashing codes, while this is not considered in CTM.

As for the speed issue, the major off-line computational costs of the proposed method come from LDA and the latent factor model parts. For LDA, we can use parallel LDA (e.g., [1]) to accelerate the computational speed. For optimizing the unified framework, it normally converges within 20 iterations and only some simple updates are necessary in each iteration. For the more important issue of calculating hashing codes of the query examples for similarity search, it is very efficient since only some linear transformations are necessary.

4. EXPERIMENTS

This section presents an extensive set of experiments to demonstrate the advantages of the proposed research.

4.1 Datasets

A set of datasets are utilized in evaluation as follows:

1. *ReutersV1* (Reuters-Volume I): This dataset contains over 800,000 manually categorized newswire stories [18]. There are in total 126 tags associated with this dataset. A subset of 365001 documents of ReutersV1 is used in our experiment. 328501 documents are randomly selected as the training data, while the remaining 36500 documents are used as testing queries. 106 tags are selected for training and 20 for testing.

2. *Reuters* (Reuters21578)[2] is a collection of documents that appeared on Reuters newswire in 1987. It contains 21578 documents, and 135 tags/categories. In our experiments, documents corresponding to the top 57 categories are kept[3], with approximately 10376 documents. 9339 documents are randomly chosen as the training set, while 1037 for testing. 47 tags are utilized in training and 10 left for testing.

3. *20Newsgroups*[4] corpus is collected and originally used for document categorization in [17]. We use the popular '18828' version which contains 18828 documents. The data is organized into 20 different newsgroups, each corresponding to a different topic. Since some of the newsgroups are very closely related to each other (*e.g.* comp.sys.ibm.pc.hardware / comp.sys.mac.hardware), while others are highly unrelated (*e.g.* misc.forsale / soc.religion.christian). Therefore, we partition these documents according to subject matter into 6 categories, which are denoted as 6 different tags. 16946 documents are randomly chosen

[2]http://daviddlewis.com/resources/textcollections/reuters21578/.
[3]we removed the tags which only have a limited number of examples.
[4]http://people.csail.mit.edu/jrennie/20Newsgroups/

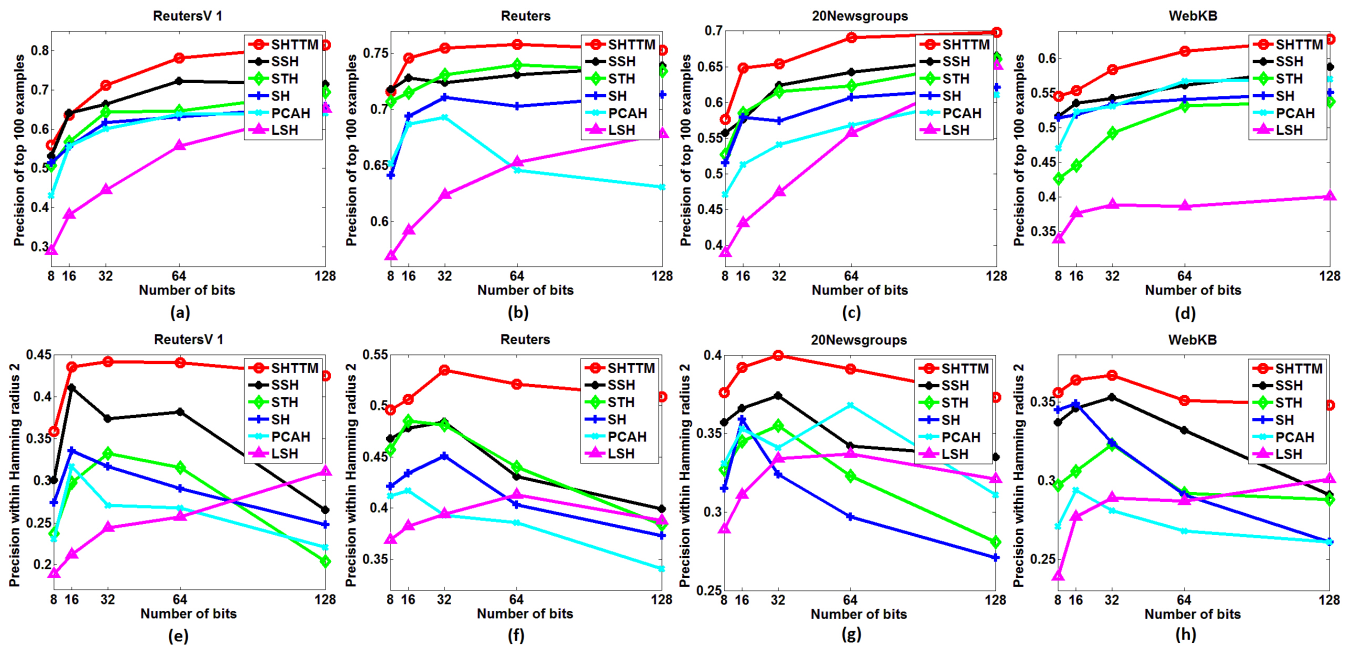

Figure 1: Results of precision of top 100 examples and Precision within Hamming radius 2 on four datasets with different hashing bits. (a)-(d): Precision of the top 100 retrieved examples. (e)-(h): Precision of the retrieved examples within Hamming radius 2.

for training and the rest 1882 documents are used for testing. 3 tags are utilized in training process and 3 for testing.

4. $WebKB$[5] consists of 6883 webpages, collected from four universities, and is divided into 7 categories/tags. 90% documents (6195) are randomly selected as training data, while the remaining (688) documents are used for testing. 4 tags are used in training while 3 for testing.

In all datasets, term frequency (i.e. tf) features are used as content features and also used for learning topic distributions. Note that tags in each dataset are divided into two groups, one set is used only in training and the other is treated as ground truth only for testing.

4.2 Evaluation Metrics

To conduct similarity search, each document in the testing set is used as a query document to search for similar documents in the corresponding training set based on the hamming distance of their hashing codes. The performance is measured with standard information retrieval performance metrics: precision as the ratio of the number of retrieved relevant documents to the number of all retrieved documents and recall as the ratio of the number of retrieved relevant documents to the number of all relevant documents. The performance is averaged over all test queries in the dataset.

There are several methodologies to determine whether a retrieved document is relevant to the given query document. In SH [34], the k closest documents in the original feature space are considered as the relevant documents. In STH [39] and CHMIS [38], the documents with the same tag as the query document are considered as the most relevant ones. The former metric is not suitable since the similarity in the original feature space may not well reflect the document semantic similarity (as discussed in section 3.3). The latter metric is also questionable since documents with more than one tag are discarded in STH and CHMIS, while documents are allowed to have multiple tags in our experiments. Therefore, we adopt a similar metric as in [20] and [31] in our experiments. In particular, a retrieved document that shares any common test tag with the query document is regarded as a relevant document.

4.3 Experiment Settings

The proposed SHTTM approach is compared with five different methods on these datasets such as Semi-Supervised Hashing (SSH) [31], Self Taught Hashing (STH) [39], Spectral Hashing (SH) [34], PCA Hashing (PCAH) [19], and Latent Semantic Hashing (LSH) [7] by using the evaluation metric described above.

The parameters α, γ and λ in SHTTM are tuned by 3-fold cross validation on the training set through the grid $\{0.01, 0.1, 1, 10, 100\}$. The number of nearest neighbors is tuned to be 7 when constructing the graph Laplacians for STH in all experiments. For LSH, we randomly select projections from a Gaussian distribution with zero-mean and identity covariance to construct the hash tables. For SSH, we sample $2k$ random points from the training set to construct the pairwise constraint matrix. We evaluate the performance of different methods by varying the number of hashing bits in the range of $\{8, 16, 32, 64, 128\}$ and calculate the average result by repeating each experiment 10 times.

[5]CMU world wide knowledge base (WebKB) project. Available at http://www.cs.cmu.edu/ WebKB/.

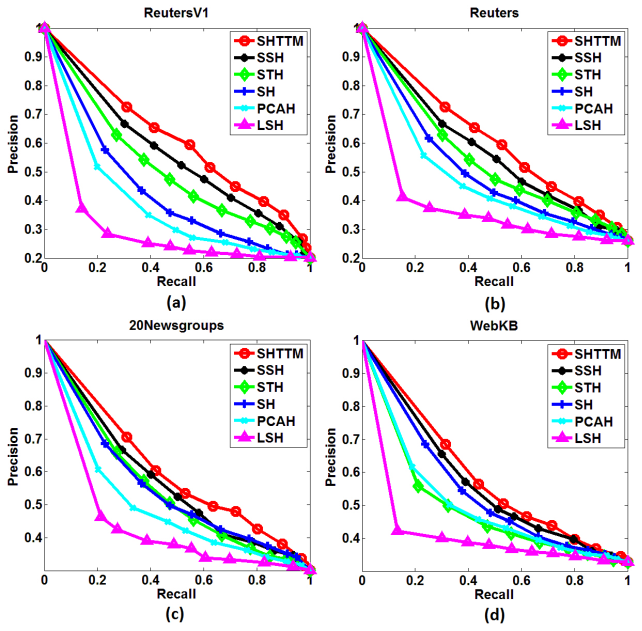

Figure 2: Results of Precision-Recall curve with 32 hashing bits on four datasets.

4.4 Results and Discussions

Four sets of experiments are conducted on each dataset to measure the performance of the proposed SHTTM approach and the five alternative methods to answer the following questions: (1) Whether SHTTM can outperform other methods in high-precision results such as the precision for the top 100 retrieved documents based on the hamming distance ranking and the precision for retrieved documents within a fixed hamming distance 2; (2) How does the SHTTM compare with other methods with recall-related metrics as the precision-recall curve? In particular, for each query document, we vary the number of retrieved documents from 0 to the number of all training documents while fixing the number of hashing bits to 32; (3) How do the two components of the SHTTM approach work that only utilize the tag information or the topic modeling information? These two variants are compared with corresponding baseline methods and the SHTTM approach that combines both tag and topic modeling information; (4) Whether SHTTM has robust performance with respect to different values of meta model parameters.

In the first set of experiments, we report the precision for the top 100 retrieved documents with different numbers of hashing bits in Fig.1(a)-(d) and Table 2. The precisions for the retrieved documents within hamming radius 2 are shown in Fig.1(e)-(h). From these comparison results, it can seen that SHTTM gives the overall best performance among all six hashing methods on all four datasets.

In Fig.1(e)-(h), the precision of most compared methods decreases when the number of hashing bits increases from 16 to 128 This is because when using longer hashing bits, the Hamming space becomes increasingly sparse and very few data points fall within the Hamming ball of radius 2, resulting in even queries with precision 0. Similar behavior is also observed in [20] and [31]. In this situation, the precision results of top 100 documents from Fig.1(a)-(d) provide better performance measurement, while the precision results of SHTTM are still consistently better than other methods.

In the second set of experiments, the precision-recall curves of different methods with 32 hashing bits on different datasets are reported in Fig.2. It can be seen that among all of these comparison methods, SHTTM shows the best performance.

From these figures, we can see that LSH does not perform well in most cases, especially its precision of the top 100 documents is not satisfactory. This is because LSH method is data-oblivious and may lead to inefficient codes in practice as also observed in [22] and [34]. For methods SH and STH, although these methods try to preserve the similarity between documents in their learned hashing codes, they do not utilize the supervised information contained in tags. Moreover, the similarity matrices in both methods are computed from the original keyword feature space, which may not fully reflect the semantic similarity between documents that goes beyond keyword matching. Therefore, the SHTTM method substantially outperforms these two methods by leveraging tag information and topic modeling. SSH achieves better results than SH and STH due to the incorporation of pairwise similarity constraints. However, as pointed out in Section 2.2, these coarse pairwise constraints generated from tags may lose detailed tag information and may not be reliable. On the other hand, the tag information is fully exploited in the SHTTM approach via modeling the semantic correlation between tags and hashing codes through a latent factor model and thus SHTTM generates higher quality hashing codes than SSH.

In the third set of experiments, the effectiveness of the two components of the proposed SHTTM such as tag modeling and topic modeling are evaluated separately. In particular, the component of tag modeling is obtained by setting γ to 0 in Eqn.7, which means we do not utilize topic models. This method is named SHTTM-TagOnly in our experiment. SHTTM-TagOnly is compared with SSH which incorporates summary tag information as pairwise constraints into learning hashing codes. the component of topic modeling is obtained by setting the tag matrix $\boldsymbol{C} = \boldsymbol{0}$ in Eqn.7, which means we do not utilize any tag information in learning hashing codes. This method is called SHTTM-TopicOnly and it is compared with STH that calculates the document similarity in the original keyword feature space.

The precision results of top 100 retrieved documents of SHTTM-TagOnly, SHTTM-TopicOnly, SHTTM and the corresponding baseline methods are shown in Fig.3 when different numbers of hashing bits are used. The experimental results in Fig.3 indicate that the modeling of tag information and utilizing topic modeling are beneficial for improving the effectiveness of hashing separately as SHTTM-TagOnly is more effective than SSH while SHTTM-TopicOnly is more effective than STH. In particular, the SHTTM-TagOnly method incorporates the complete tag information into learning hashing codes via a latent factor model, while SSH only utilizes the partial tag information in pairwise constraints. The SHTTM-TopicOnly method preserves the document similarity in the topic/semantic, while the original keyword feature space used by STH may not fully reflect the semantic relationships that go beyond keyword matching. Finally, combining these two components together, the proposed SHTTM achieves even better performance, which is consistent with our expectation.

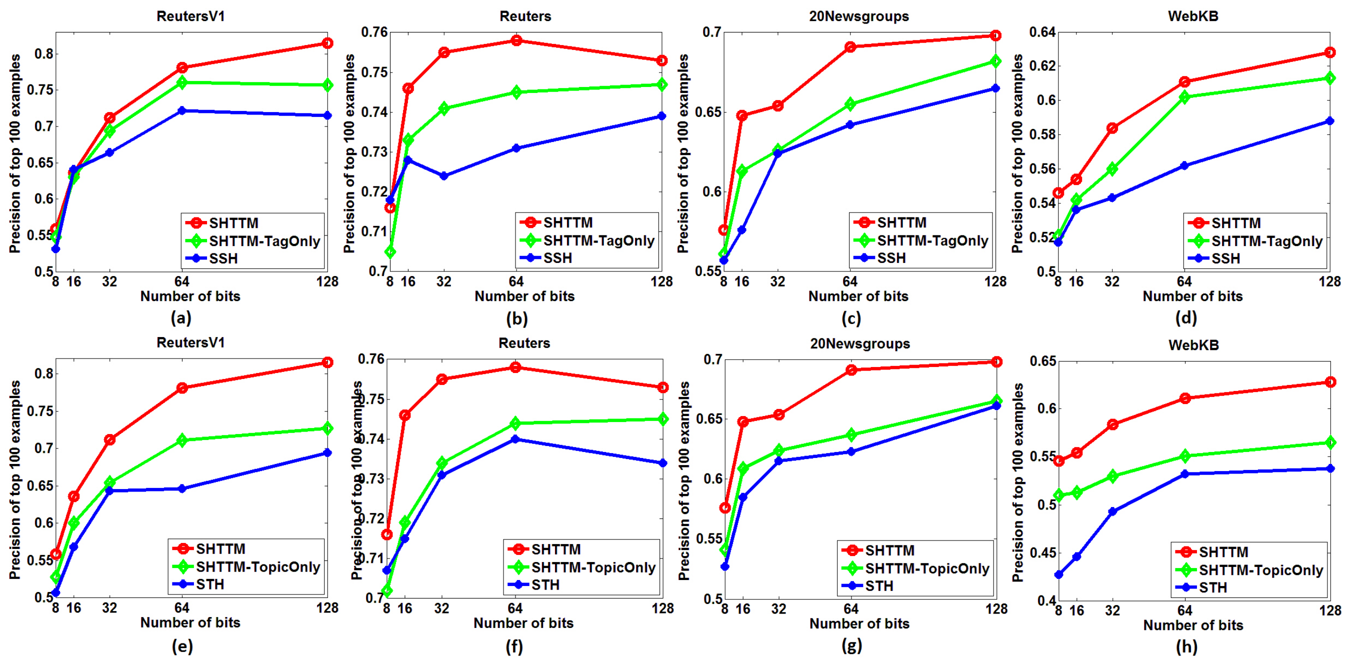

Figure 3: Precision of the top 100 retrieved documents of SHTTM-TagOnly, SHTTM-TopicOnly and SHTTM on four datasets with different hashing bits. (a)-(d): Comparison results of using tag model only. (e)-(h): Comparison results of using topic model only.

The fourth set of experiments study the performance of SHTTM with respect to the meta parameters α, γ and λ. To prove the robustness of the proposed method, we conduct parameter sensitivity experiments on all datasets. In each experiment, we tune only one parameter from $\{0.5, 1, 2, 4, 8, 16, 32, 128\}$, while fixing the other two to the optimal values obtained from the first set of experiments. We report the results on $ReutersV1$ and $20Newsgroups$ in Fig.4. It is clear from these experimental results that the performance of SHTTM is relatively stable with respect to α, γ and λ. We also observe similar results of the proposed method in the other two datasets. But due to the limit of space, they are not presented here.

The prediction procedure for generating hashing codes is very fast. The linear hashing function allows a quick mapping of a query document from the original high-dimensional feature space to the low-dimensional space of hashing codes. We implement our method using Matlab on a PC with Intel Duo Core i5-2400 CPU 3.1GHz and 4GB RAM. It takes 0.0002 second average per query document for the prediction procedure (Table 1) on all datasets.

5. CONCLUSIONS

Similarity search has become an important technique in many information retrieval applications such as search and recommendation. Many applications with similarity search often involve a large amount of data, which demands effective and efficient solutions. Semantic hashing has been proposed for the problem to map data examples like documents in a high-dimensional space (e.g., a vector space of keywords in the vocabulary) into a low-dimensional binary vector space, which at the same time preserves the semantic relationship of the data examples as much as possible. Valuable prior research has been conducted in this direction for learning hashing codes and mapping function with techniques such as unsupervised learning and supervised learning. However, most existing research on semantic hashing is only based on content similarity computed in the original keyword feature space. Moreover, tag information is not fully utilized in existing research, although they are often available in many information retrieval applications.

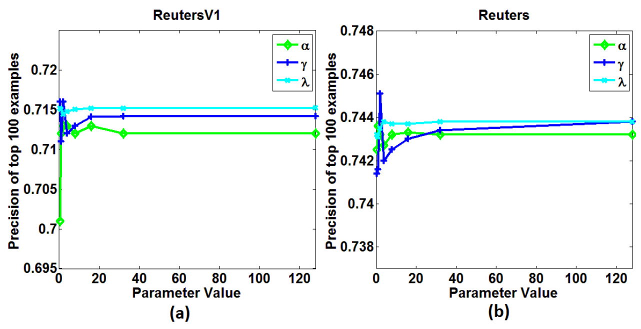

Figure 4: Parameter Sensitivity for α, γ and λ. Results of precision of the top 100 retrieved documents with 32 hashing bits.

This paper proposes novel research for semantic hashing that jointly considers tag information and semantic similarity in generating hashing codes. The proposed research utilizes topic modeling to explore semantic similarity between documents that goes beyond keyword matching. The new SHTTM approach incorporates two components as tag consistency and topic consistency together into a joint objective function for learning desired tag representation and topic representation simultaneously. An iterative coordinate descent method is proposed to obtain the optimal hashing codes by solving the objective function. An extensive set of experiments has shown that the proposed new research can generate more accurate hashing codes than several other state-of-the-art hashing methods. In particular, the experiments clearly demonstrate the benefits of utilizing tag information and topic modeling information separately, and further show that the joint approach generates the best results by combining both two types of information.

There are several possible directions to explore in the future research. For example, the research in this paper only utilizes a fixed number of topics, which may not be optimal for different types of datasets. We plan to explore new research for automatically adjusting the number of topics with either model selection methods or a nonparametric Bayesian approach.

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] A. Ahmed, M. Aly, J. Gonzalez, S. Narayanamurthy, and A. J. Smola. Scalable inference in latent variable models. In *WSDM*, pages 123–132, 2012.

[2] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.

[3] D. Blei and J. Lafferty. Topic models. *Text Mining: Theory and Applications*, 2009.

[4] D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003.

[5] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms.* The MIT press, 2001.

[6] J. S. Culpepper and A. Moffat. Efficient set intersection for inverted indexing. *ACM Trans. Inf. Syst.*, 29(1):1, 2010.

[7] 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.

[8] S. C. Deerwester, S. T. Dumais, T. K. Landauer, G. W. Furnas, and R. A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

[9] A. Gionis, P. Indyk, and R. Motwani. Similarity search in high dimensions via hashing. In *VLDB*, pages 518–529, 1999.

[10] G. E. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.

[11] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[12] T. Hofmann. Probabilistic latent semantic indexing. In *SIGIR*, pages 50–57, 1999.

[13] T. Hofmann. Collaborative filtering via gaussian probabilistic latent semantic analysis. In *SIGIR*, pages 259–266, 2003.

[14] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *ICDM*, pages 263–272, 2008.

[15] W. Kong and W.-J. Li. Isotropic hashing. In *NIPS*, pages 1655–1663. 2012.

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

[17] K. Lang. Newsweeder: Learning to filter netnews. In *ICML*, pages 331–339, 1995.

[18] 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.

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

[20] W. Liu, J. Wang, R. Ji, Y.-G. Jiang, and S.-F. Chang. Supervised hashing with kernels. In *CVPR*, pages 2074–2081, 2012.

[21] W. Liu, J. Wang, S. Kumar, and S.-F. Chang. Hashing with graphs. In *ICML*, pages 1–8, 2011.

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

[23] G. Salton. Developments in automatic text retrieval. *Science*, 253(5023):974–980, August 1991.

[24] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Inf. Process. Manage.*, 24(5):513–523, 1988.

[25] L. Si and R. Jin. Flexible mixture model for collaborative filtering. *In ICML*, pages 704–711, 2003.

[26] F. Silvestri and R. Venturini. Vsencoding: efficient coding and fast decoding of integer lists via dynamic programming. In *CIKM*, pages 1219–1228, 2010.

[27] B. Stein. Principles of hash-based text retrieval. In *SIGIR*, pages 527–534, 2007.

[28] V. Vapnik. *The nature of statistical learning theory.* Springer Verlag, 2000.

[29] C. Wang and D. M. Blei. Collaborative topic modeling for recommending scientific articles. In *KDD*, pages 448–456, 2011.

[30] J. Wang, S. Kumar, and S.-F. Chang. Sequential projection learning for hashing with compact codes. In *ICML*, pages 1127–1134, 2010.

[31] J. Wang, S. Kumar, and S.-F. Chang. Semi-supervised hashing for large-scale search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(12):2393–2406, 2012.

[32] 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.

[33] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *VLDB*, pages 194–205, 1998.

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

[35] H. Xia, P. Wu, S. C. H. Hoi, and R. Jin. Boosting multi-kernel locality-sensitive hashing for scalable image retrieval. In *SIGIR*, pages 55–64, 2012.

[36] X. Yi and J. Allan. Evaluating topic models for information retrieval. In *CIKM*, pages 1431–1432, 2008.

[37] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. In *NIPS*, 2004.

[38] D. Zhang, F. Wang, and L. Si. Composite hashing with multiple information sources. In *SIGIR*, pages 225–234, 2011.

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

[40] J. Zobel and A. Moffat. Inverted files for text search engines. *ACM Comput. Surv.*, 38(2), 2006.