

2011

Composite Hashing with Multiple Information Sources

Dan Zhang

Fei Wang

Luo Si

Purdue University, lsi@cs.purdue.edu

Follow this and additional works at: <http://docs.lib.purdue.edu/ccpubs>

 Part of the [Engineering Commons](#), [Life Sciences Commons](#), [Medicine and Health Sciences Commons](#), and the [Physical Sciences and Mathematics Commons](#)

Zhang, Dan; Wang, Fei; and Si, Luo, "Composite Hashing with Multiple Information Sources" (2011). *Cyber Center Publications*. Paper 415.

<http://docs.lib.purdue.edu/ccpubs/415>

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.

Composite Hashing with Multiple Information Sources

Dan Zhang
Computer Science
Department
Purdue University,
West Lafayette, IN 47907, US
zhang168@cs.purdue.edu

Fei Wang
IBM T. J. Watson Research
Lab
Hawthorne, NY 10532
fwang@us.ibm.com

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

ABSTRACT

Similarity search applications with a large amount of text and image data demands an efficient and effective solution. One useful strategy is to represent the examples in databases as compact binary codes through semantic hashing, which has attracted much attention due to its fast query/search speed and drastically reduced storage requirement. All of the current semantic hashing methods only deal with the case when each example is represented by one type of features. However, examples are often described from several different information sources in many real world applications. For example, the characteristics of a webpage can be derived from both its content part and its associated links.

To address the problem of learning good hashing codes in this scenario, we propose a novel research problem – Composite Hashing with Multiple Information Sources (CHMIS). The focus of the new research problem is to design an algorithm for incorporating the features from different information sources into the binary hashing codes efficiently and effectively. In particular, we propose an algorithm CHMIS-AW (CHMIS with Adjusted Weights) for learning the codes. The proposed algorithm integrates information from several different sources into the binary hashing codes by adjusting the weights on each individual source for maximizing the coding performance, and enables fast conversion from query examples to their binary hashing codes. Experimental results on five different datasets demonstrate the superior performance of the proposed method against several other state-of-the-art semantic hashing techniques.

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

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

SIGIR '11, July 24–28, 2011, Beijing, China.

Copyright 2011 ACM 978-1-4503-0757-4/11/07 ...\$10.00.

General Terms

Algorithms, Performance, Experimentation

Keywords

Composite Hashing, Semantic Hashing, Multiple Information Sources

1. INTRODUCTION

The explosive growth of the internet has generated a huge amount of data such as documents, images and videos. As the data sizes increase, the density of similar objects in the data space also increases. Therefore, nearest neighbor methods for similarity search applications tend to be more reliable than before. However, two key problems for using nearest neighbor search in large datasets are: (1) *Storage challenge*. How to store the training data efficiently? (2) *Retrieval challenge*. How to retrieve the desired data efficiently and effectively? It is clear that the traditional methods, such as TF-IDF for document representations [34, 35], which compare word count vectors, are difficult to be directly used for large datasets, since they need to save the original examples and the computational cost of dealing with floating/integer features is often too high.

A clever way of solving these two challenges is through semantic hashing [17, 33, 47]. By using semantic hashing algorithms, each example in the database is re-represented by a compact binary code, which preserves its semantic meanings in the original feature space. These hashing methods also provide a way to efficiently transform query examples into the corresponding hashing codes. Then, the retrieving process can be simply done by selecting examples within a small Hamming distance of the codes for the query example. This method addresses the two challenges in the following ways: (1) By mapping examples in the database to a low dimensional binary space, it is much more efficient in terms of the storage cost, compared with saving the original examples. (2) The retrieval speed is fast, since an efficient scheme for mapping the query examples to their hashing codes is provided, and the similarity computation can be simply done by using bit XOR operation and counting the number of ‘0’ bits in the low dimensional space. This process is very fast, and nowadays even an ordinary laptop is capable of doing millions of Hamming distance computation in a short time.

Previous semantic hashing methods are successful in addressing the two challenges. However, they do not address the case when examples are described from several different sources. Actually, in many real-world applications, it is

often true that examples are derived from several different sources, and therefore are represented by multiple sets of features. For example, in web mining applications, each webpage has disparate descriptions, textual content, in-bound and out-bound links, etc. In image retrieval, each picture can be described by different kinds of features, such as the SIFT features [27], RGB features, texture features, etc. Different sets of features could have different statistical properties. Therefore, designing a hashing algorithm for examples from multiple information sources is necessary. The main challenge for this task is how to incorporate different sets of features together into one set of binary hashing codes.

To address this task, we propose a novel research problem: Composite Hashing with Multiple Information Sources (CHMIS). The basic objective of CHMIS is to design some efficient and effective hashing algorithms that can encode the examples described from several different information sources. An intuitive way of designing a CHMIS method is to simply concatenate the features from several different sources, treat them as if they were from one information source, and apply the previous hashing method to these examples. However, by doing so, the different statistical properties from different individual sources may be lost. This paper proposes an elegant method – CHMIS-AW (CHMIS with Adjusted Weights) to intelligently integrate information from different sources. In particular, CHMIS-AW is an iterative method which preserves the semantic similarities between training examples, and ensures the consistency between the hashing codes and the corresponding hashing functions designed for different information sources. Furthermore, the importance of each individual source is represented as a convex combination coefficient and can be automatically learned through a joint optimization procedure. Our experiments on five real world datasets with different information sources show that the proposed method outperforms state-of-the-art methods substantially.

2. RELATED WORK

Efficiency is a critical issue for many information retrieval applications with a large number of text documents, images or videos. For traditional ad-hoc text search with relatively short user queries, different types of structures and operations of inverted indexing have been proposed [13, 39, 48, 54]. Distributed indexing [2, 3] has also been explored when there is sufficient computing resource. On the other side, similarity search with documents, images, or other entities are typically represented as feature vectors in a space of more than thousands of dimensions [25, 28], which demands different solutions especially when only limited computing resource is available.

The traditional similarity search is conducted based on these feature vectors by space partitioning index structures, like TF-IDF methods [34, 35], KD-tree, or data partitioning index structures, say R-tree [12]. However, when the dimensionality of feature space is too high, traditional similarity search may fail to work efficiently [46]. Semantic hashing [33] is used in the case when the requirement for the exactness of the final results is not high, and the similarity search in the original high dimensional space is not affordable. More precisely, semantic hashing methods try to represent the whole datasets by using a fixed small number of binary bits so that the queries can be answered in a short time (virtually constant time) [41], with some extent of pre-

cision being guaranteed. Such hashing based fast similarity search can be considered as a way to embed high dimensional feature vectors into a Hamming space, while preserving the semantic similarity relationship between examples as much as possible. This is different from traditional dimensionality reduction methods, such as Principal Component Analysis (PCA) and Latent Semantic Analysis (LSI) [15, 20]. Hashing methods map original features to low dimensional binary codes, which enables the efficient search in Hamming space.

One of the most well known hashing methods is Locality-Sensitive Hashing (LSH) [1, 14]. It uses random linear projections to map close examples to similar codes. It has already been proved that the Hamming distance between different examples will asymptotically approach their Euclidean distance in the original feature space, with the increase of the hashing bits. Some LSH based methods are also used for near optimal duplicate detection. For example, min-Hash function [8–10] assigns numbers to each example, and, for two documents, the probability of being assigned the same number equals the ratio of the intersection and union of their word representations.

Besides LSH, some machine learning methods can be adapted to solve the hashing problem. The traditional dimensionality reduction methods can be adapted to solve the hashing problem via a simple thresholding [33, 45]. For example, PCA Hashing [26, 45] computes K -bit hashing codes by projecting each example to the K principal components of the training set, and then binarizing the coefficients, by setting each bit to 1 if it exceeds the median value seen for the training set, and -1 otherwise. In [33], the authors use stacked Restricted Boltzman Machine (RBM) [18, 19] to generate compact binary hashing codes, which can be considered as binarized LSI. Later, in [51], the authors further improve this method. Some traditional classification algorithms can also be adapted. For example, in [4, 37], the authors adapted Adaboost [31] to the hashing method. More precisely, they consider the similar pairs of examples as positive examples, and otherwise negative. Then, a set of classifiers are trained by AdaBoost, the output of all of these weak learners are considered as the binary hashing codes. In [43, 44], the authors solve the problem of semi-supervised semantic hashing. In the following two sections, we mainly discuss two state-of-the-art hashing methods that most related to the proposed method – Spectral Hashing [47] and Self Taught Hashing [52], which have been shown to outperform several other types of hashing methods.

2.1 Spectral Hashing (SH)

As a recently proposed method, Spectral Hashing (SH) [47] was proposed to design compact binary codes for search. This method can be considered as an extension of spectral clustering [49]. Its concrete formulation is as follows:

$$\begin{aligned} \min \sum_{i,j} \mathbf{S}_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \\ \text{s.t. } \mathbf{y}_i \in \{-1, 1\}^K, \sum_i \mathbf{y}_i = 0, \frac{1}{n} \sum_i \mathbf{y}_i \mathbf{y}_i^T = \mathbf{I}, \end{aligned} \quad (1)$$

where, \mathbf{S}_{ij} is the similarity measure between the example i and example j , \mathbf{y}_i is the hashing code for the i -th example, K is the number of bits. The basic motivation of this formulation is to preserve the original similarity relationship between examples in the Hamming space. However, solving

this above problem is NP hard. After relaxing the discrete constraints to continuous, it can be solved using traditional spectral analysis. Then, the optimal hashing code can be simply obtained by binarizing the solution of the relaxed optimization problem. Moreover, this formulation can be generalized to enable the effective computation for out of sample examples/queries, as introduced in [47].

SH tries to keep the similarity relationships, which are defined in the original feature space, between examples in the hashing codes. However, it is hard to find a good similarity measure which can consider the consistency of the different sources. Actually, some previous works demonstrate an improvement for considering the consistency in multiple source problems [32, 40, 50], in which the authors designed a classifier on each view, and require the outputs from different views should be consistent and not deviate too much.

2.2 Self Taught Hashing (STH)

Self Taught Hashing (STH) [52] can be considered as an extension to SH. To obtain the hashing codes for the training set and the hashing function for efficiently mapping the query examples, STH uses two steps, an unsupervised step and a supervised step. In the unsupervised step, the authors construct a k -nearest-neighbor graph for the given dataset, embed the examples into a K (K is still the number of bits) dimensional space through spectral analysis, and obtain the binary codes for each example through thresholding. In the supervised step, K SVM classifiers are trained based on the training examples, and their binary hashing codes learned from the previous step are used as labels. The hashing codes for the query example can then be obtained through a classification problem and thresholding by using the K classifiers.

Compared with SH, when dealing with query examples, STH does not assume that data are uniformly distributed in a hyper-rectangle, which is restrictive. The maximum margin principle enables a good generalization ability. However, it still cannot avoid the same consistency problem of the similarity measures for different sources as in SH. Different from their method, the method proposed in this paper combines both the unsupervised and supervised steps for intelligently integrating features from multiple sources.

3. PROBLEM STATEMENT

In CHMIS, we are given a set of n training examples from M information sources, represented as: $\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_n^{(i)}\} \in \mathbf{R}^{d_i \times 1}$, $i \in \{1, 2, \dots, M\}$, where d_i is the dimensionality of the i -th information source. The main objective of CHMIS is to find the optimal K binary hashing bits $\mathbf{Y} \in \{-1, 1\}^{K \times n}$ for these training examples, as well as a hashing function $f(\bullet)$ that deals with the query (out-of-sample) examples. The coding optimization criterion is to identify neighbors of the query examples in the training set accurately in a short time. For convenience, throughout this paper, the hashing codes for the i -th training example, i.e., the i -th column of \mathbf{Y} , is denoted as \mathbf{y}_i . The hashing codes for the p -th hashing bit, which is the p -th row of \mathbf{Y} , is denoted as \mathbf{Y}^p .

4. COMPOSITE HASHING WITH MULTIPLE INFORMATION SOURCES WITH ADJUSTED WEIGHTS

From the previous presentation, we can see that CHMIS is an important research problem. To solve it, this pa-

per presents a novel optimization formulation – CHMIS-AW (CHMIS with Adjusted Weights).

The main objective of CHMIS-AW is to obtain hashing codes for training examples and to learn the hashing function simultaneously by preserving the similarity relationships between training examples in the original feature space, and generating a consistent hashing function from different information sources. In particular, the objective function of CHMIS-AW is composed of two parts: The first part is a similarity preservation term, which tries to preserve the similarity relationship in the original feature space using the learned hashing codes. The second part ensures the consistency between the learned hashing codes and the corresponding hashing function designed on multiple sources. In the following, we will first introduce how to construct the two parts respectively. Then the joint optimization problem and the corresponding optimization strategy will be provided. Finally, we will discuss the distinctions of the proposed method and some related ones.

4.1 Similarity Preservation

One of the key goals in most state-of-the-art hashing methods, such as [33, 47], is to seek for compact binary codes so that the Hamming distance between codewords correlates with semantic similarity. This indicates that similar data points should be mapped to similar codes within a short Hamming distance. In SH, the authors demonstrated that the problem of finding such an optimal set of codes is closely related to the graph partition methods. They proposed to achieve the goal by using the spectral method.

Different from the previous hashing problems, in CHMIS-AW, we need to deal with multiple information sources. To measure the similarity between examples represented by the binary hashing codes, one natural way is to measure the similarity quantity on each individual source and sum them together as follows:

$$\sum_{t=1}^M \sum_{i,j=1}^n \mathbf{S}_{ij}^{(t)} \|\mathbf{y}_i - \mathbf{y}_j\|^2. \quad (2)$$

Here, $\mathbf{S}_{n \times n}^{(t)}$ is the affinity matrix defined on the t -th source. 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 on the information source.

There are many different ways of defining the affinity matrix $\mathbf{S}^{(t)}$. In [47], the author used the global similarity structure of all document pairs, while in [52], the local similarity structure, i.e., k -nearest-neighborhood, is used. In this paper, we use the local similarity, due to its nice property in many data mining and information retrieval applications [28]. In particular, the corresponding weights are computed by Gaussian functions, i.e.,

$$\mathbf{S}_{ij}^{(t)} = \begin{cases} e^{-\frac{\|\mathbf{x}_i^{(t)} - \mathbf{x}_j^{(t)}\|^2}{\delta_{ij}^2}} & \text{if } \mathbf{x}_i^{(t)} \in N_k(\mathbf{x}_j^{(t)}) \text{ or } \mathbf{x}_j^{(t)} \in N_k(\mathbf{x}_i^{(t)}) \\ 0 & \text{otherwise} \end{cases}$$

The variance δ_{ij} is determined automatically by local scaling [49], and $N_k(\mathbf{x})$ represents the set of k -nearest-neighbors of the example \mathbf{x} .

By introducing a diagonal $n \times n$ matrix $\mathbf{D}^{(t)}$ for each individual source, whose entries are given by $\mathbf{D}_{ii}^{(t)} = \sum_{j=1}^n \mathbf{S}_{ij}^{(t)}$,

Eq.(2) can be rewritten as:

$$\text{tr} \left(\mathbf{Y}^T \sum_{t=1}^M (\mathbf{D}^{(t)} - \mathbf{S}^{(t)}) \mathbf{Y} \right) = \text{tr} \left(\mathbf{Y}^T \sum_{t=1}^M \mathbf{L}^{(t)} \mathbf{Y} \right), \quad (3)$$

where $\mathbf{L}^{(t)}$ is the graph Laplacian [11] defined on the t -th source, and $\text{tr}(\bullet)$ is the trace function. Furthermore, similar to [52], we can replace the graph Laplacian with the normalized graph laplacian due to its superior performance [11, 38] indicated by:

$$\tilde{\mathbf{L}}^{(t)} = (\mathbf{D}^{(t)})^{-\frac{1}{2}} \mathbf{L}^{(t)} (\mathbf{D}^{(t)})^{-\frac{1}{2}}. \quad (4)$$

Then the objective function that needs to be minimized turns to:

$$\Omega(\mathbf{Y}) = \text{tr} \left(\mathbf{Y}^T \sum_{t=1}^M \tilde{\mathbf{L}}^{(t)} \mathbf{Y} \right). \quad (5)$$

By minimizing this term, the similarity between different examples can be preserved in the learned hashing codes.

4.2 Consistency

In Section 4.1, our focus is to find the optimal binary hashing matrix \mathbf{Y} that preserves the data similarities in the original feature space. However, this is only a transductive formulation, i.e., \mathbf{Y} cannot be generalized to query examples directly. It is true that we can use the Nyström method [5] to deduct the hashing codes for the query examples from the hashing codes in the training set and the similarity matrix between the query and the training examples, but this operation is as expensive as doing exhaustive nearest neighbor search. We solve this problem by first introducing a linear hash function¹, which is represented as: $f(\mathbf{x}_i) = \sum_{t=1}^M \alpha_t (\mathbf{W}^{(t)})^T \mathbf{x}_i^{(t)}$, where $\mathbf{W}^{(t)} \in \mathbf{R}^{d_t \times K}$ is the weight vector for the classifier on the t -th source, and $\alpha = [\alpha_1, \dots, \alpha_M]^T$ is a M dimensional non-negative convex combination coefficient column vector that balances the outputs from each individual source, and $\sum_{t=1}^M \alpha_t = 1$.

In this way, the output of this hashing function is a convex combination [7] of the linear hash function outputs on each individual source. A constraint here is that the outputs on the training set should be as close to \mathbf{Y} as possible, i.e., the outputs of the hash functions from different sources should make an ‘‘agreement’’ on the learned binary hashing codes. The importance of each individual source on this agreement is specified by the corresponding convex combination coefficient. In this way, this methodology incorporates the different statistic properties of the different sources. The concrete formulation for the consistency part is given as follows:

$$C_2 \sum_{i=1}^n \|\mathbf{y}_i - \sum_{t=1}^M \alpha_t (\mathbf{W}^{(t)})^T \mathbf{x}_i^{(t)}\|^2 + \sum_{t=1}^M \|\mathbf{W}^{(t)}\|^2, \quad (6)$$

where, the regularization term $\sum_{t=1}^M \|\mathbf{W}^{(t)}\|^2$ is introduced to avoid overfitting [42]. $\sum_{i=1}^n \|\mathbf{y}_i - \sum_{t=1}^M \alpha_t (\mathbf{W}^{(t)})^T \mathbf{x}_i^{(t)}\|^2$ is a loss function, which measures the difference between the hashing codes and the outputs of the hash functions. C_2 is a trade-off parameter, balancing the loss function and the regularization term.

¹Although [30, 43, 44, 52] use linear classifiers, as they have claimed, the kernel methods [36] can be easily incorporated into their formulations. So is the proposed method.

For convenience, we introduce the following concatenate matrix:

$$\begin{aligned} \tilde{\mathbf{W}} &= [(\mathbf{W}^{(1)})^T, (\mathbf{W}^{(2)})^T, \dots, (\mathbf{W}^{(M)})^T]^T; \\ \tilde{\mathbf{x}}_i^{(t)} &= [\mathbf{0}, \dots, (\mathbf{x}_i^{(t)})^T, \dots, \mathbf{0}]^T. \end{aligned} \quad (7)$$

Here, after this concatenation, the new designed $\tilde{\mathbf{W}}$ is a $(d_1 + d_2 + \dots + d_M) \times K$ dimensional matrix, and $\tilde{\mathbf{x}}_i^{(t)}$ is a one-dimensional column vector with only from the $(d_1 + d_2 + \dots + d_{p-1} + 1)$ -th to $(d_1 + d_2 + \dots + d_p)$ -th elements being nonzero. It is clear that $(\mathbf{W}^{(t)})^T \mathbf{x}_i^{(t)} = \tilde{\mathbf{W}}^T \tilde{\mathbf{x}}_i^{(t)}$. Therefore, Eq.(6) can be simplified as:

$$\begin{aligned} J(\mathbf{Y}, \tilde{\mathbf{W}}, \alpha) &= C_2 \sum_{i=1}^n \|\mathbf{y}_i - \sum_{t=1}^M \alpha_t (\tilde{\mathbf{W}})^T \tilde{\mathbf{x}}_i^{(t)}\|^2 + \|\tilde{\mathbf{W}}\|^2 \\ &= C_2 \|\mathbf{Y} - \sum_{t=1}^M \alpha_t (\tilde{\mathbf{W}})^T \tilde{\mathbf{X}}^{(t)}\|^2 + \|\tilde{\mathbf{W}}\|^2, \end{aligned} \quad (8)$$

where $\tilde{\mathbf{X}}^{(t)} = [\tilde{\mathbf{x}}_1^{(t)}, \tilde{\mathbf{x}}_2^{(t)}, \dots, \tilde{\mathbf{x}}_n^{(t)}]$.

4.3 Overall Objective

The overall objective function combines the similarity preservation part given in Eq.(5) and the consistency part in Eq.(8) as follows:

$$\begin{aligned} \min_{\mathbf{Y}, \tilde{\mathbf{W}}, \alpha} \quad & T(\mathbf{Y}, \tilde{\mathbf{W}}, \alpha) \\ \text{s.t.} \quad & \mathbf{Y} \in \{-1, 1\}^{K \times n}, \\ & \mathbf{Y}\mathbf{1} = \mathbf{0}, \quad \mathbf{Y}\mathbf{Y}^T = \mathbf{I}, \quad \alpha^T \mathbf{1} = 1, \quad \alpha \succeq 0, \end{aligned} \quad (9)$$

where $T(\mathbf{Y}, \tilde{\mathbf{W}}, \alpha) = \text{tr}(C_1 \mathbf{Y}^T \sum_{t=1}^M \tilde{\mathbf{L}}^{(t)} \mathbf{Y}) + C_2 \|\mathbf{Y} - \sum_{t=1}^M \alpha_t (\tilde{\mathbf{W}})^T \tilde{\mathbf{X}}^{(t)}\|^2 + \|\tilde{\mathbf{W}}\|^2$. Here, $\mathbf{Y}\mathbf{1} = \mathbf{0}$ requires each bit to fire 50% of the time (with equal probability as positive or negative), and the constraint $\mathbf{Y}\mathbf{Y}^T = \mathbf{I}$ requires the bits to be uncorrelated.

This is a hard optimization problem because of the discrete constraints. We propose to relax this constraint and drop the constraint $\mathbf{Y}\mathbf{1} = \mathbf{0}$ first. However, even after the relaxation, the objective function $T(\mathbf{Y}, \tilde{\mathbf{W}}, \alpha)$ is still non-convex with respect to $\mathbf{Y}, \tilde{\mathbf{W}}, \alpha$ jointly, which makes it difficult to solve.

To solve this problem, we will show the optimal solution of $\tilde{\mathbf{W}}$ has a closed form solution with respect to \mathbf{Y} and α . For notational convenience, we rewrite $\sum_{t=1}^M \alpha_t \tilde{\mathbf{X}}^{(t)}$ as $\tilde{\mathbf{X}}_\alpha$. Then, the optimal $\tilde{\mathbf{W}}$ can be determined by solving the following optimization problem:

$$\min_{\tilde{\mathbf{W}}} C_2 \|\mathbf{Y} - (\tilde{\mathbf{W}})^T \tilde{\mathbf{X}}_\alpha\|^2 + \|\tilde{\mathbf{W}}\|^2. \quad (10)$$

This is a standard regularized least square problem [6], and its optimal solution can be obtained by: $\tilde{\mathbf{W}} = \mathbf{Q}\mathbf{Y}$, where $\mathbf{Q} = C_2 (C_2 \tilde{\mathbf{X}}_\alpha \tilde{\mathbf{X}}_\alpha^T + \mathbf{I})^{-1} \tilde{\mathbf{X}}_\alpha$. Bringing this solution back to $T(\mathbf{Y}, \tilde{\mathbf{W}}, \alpha)$, we can eliminate the optimization variable $\tilde{\mathbf{W}}$, and problem (9) can be rewritten as:

$$\begin{aligned} \min_{\mathbf{Y}, \alpha} \quad & \tilde{T}(\mathbf{Y}, \alpha) \\ \text{s.t.} \quad & \mathbf{Y}\mathbf{Y}^T = \mathbf{I}, \quad \alpha^T \mathbf{1} = 1, \quad \alpha \succeq 0, \end{aligned} \quad (11)$$

Algorithm: CHMIS-AW Training
Input:
1. A set of training examples from M information sources: $\{\mathbf{x}_1^{(i)}, \mathbf{x}_2^{(i)}, \dots, \mathbf{x}_n^{(i)}\}, i \in \{1, 2, \dots, M\}$.
2. Parameters: Similarity Preservation parameter C_1 , and Consistency trade-off parameter C_2 , as in Eq.(9).
Output: The hashing code for training examples \mathbf{Y} , hashing function weight vector $\widetilde{\mathbf{W}}$, convex combination coefficient α and median vector \mathbf{m}
Initialization:
1. Initialize $\alpha_i = \frac{1}{M}$ ($i = 1, \dots, M$)
2. Construct $\widetilde{\mathbf{X}}^{(t)}, t = 1, \dots, M$ as in Eq.(7)
3. Construct the normalized graph Laplacians $\widetilde{\mathbf{L}}^{(t)}, t = 1, \dots, M$ on each individual view as in Eq.(4).
Repeat
4. Calculate $\widetilde{\mathbf{X}}_\alpha = \sum_{t=1}^M \alpha_t \widetilde{\mathbf{X}}^{(t)}$
5. Calculate $\mathbf{Q} = C_2(C_2 \widetilde{\mathbf{X}}_\alpha \widetilde{\mathbf{X}}_\alpha^T + \mathbf{I})^{-1} \widetilde{\mathbf{X}}_\alpha$
6. Calculate the matrix $\mathbf{H}(\alpha)$ as in Eq.(12)
7. Solve the K eigenvectors $\mathbf{Y} = \{\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K\}$ corresponding to the smallest K eigenvalues of $\mathbf{H}(\alpha)$
8. Solve $\widetilde{\mathbf{W}}, \widetilde{\mathbf{W}} = \mathbf{Q}\mathbf{Y}$
9. Compute α by solving problem (13)
Until Convergence
Training Hashing Code Generation:
For $p = 1:K$
10. Get the median value for the p -th code by $\mathbf{m}_p = \text{median}(\mathbf{Y}^p)$
11. Generating the p -th hashing code by $\mathbf{Y}^p = (\mathbf{Y}^p > \mathbf{m}_p)$
End For

Table 1: Algorithm: CHMIS-AW Training

where,

$$\begin{aligned} \widetilde{T}(\mathbf{Y}, \alpha) &= \text{tr} \left(\mathbf{Y}^T (C_1 \sum_{t=1}^M \widetilde{\mathbf{L}}^{(t)} + C_2(\mathbf{I} - \mathbf{Q}^T \widetilde{\mathbf{X}}_\alpha \right. \\ &\quad \left. - \widetilde{\mathbf{X}}_\alpha^T \mathbf{Q} + \mathbf{Q}^T \widetilde{\mathbf{X}}_\alpha \widetilde{\mathbf{X}}_\alpha^T \mathbf{Q}) + \mathbf{Q}^T \mathbf{Q}) \mathbf{Y} \right) \\ &= \text{tr} \left(\mathbf{Y}^T \mathbf{H}(\alpha) \mathbf{Y} \right). \end{aligned} \quad (12)$$

Here, $\mathbf{H}(\alpha)$ is a positive semi-definite matrix, which is related to α and is introduced to simplify the formulation. \mathbf{I} is an identity matrix.

The problem (11) is still non-convex, since α and \mathbf{Y} are still coupled together. Fortunately, the problem is convex with respect to either of them, with the other one fixed, and therefore can be solved by alternative optimization with guaranteed convergence. In particular, after initializing α , the optimization problem can be solved by doing the following two steps iteratively, until convergence:

1. Fix α , optimize:

$$\min_{\mathbf{Y}} \mathbf{Y}^T \mathbf{H}(\alpha) \mathbf{Y}, \quad \text{s.t.} \quad \mathbf{Y}\mathbf{Y}^T = \mathbf{I}$$

The solution of this optimization is given by $\mathbf{Y} = [\mathbf{v}_1, \mathbf{v}_2, \dots, \mathbf{v}_K]^T$, whose columns are the K eigenvectors corresponding to the smallest eigenvalues of $\mathbf{H}(\alpha)$. Then we calculate $\widetilde{\mathbf{W}} = \mathbf{Q}\mathbf{Y}$.

2. Fix \mathbf{Y} , optimize:

$$\min_{\alpha} \|\mathbf{Y} - \mathbf{O}\alpha\|^2, \quad \text{s.t.} \quad \alpha^T \mathbf{1} = 1, \quad \alpha \succeq 0 \quad (13)$$

where $\mathbf{O} = [\mathbf{o}^{(1)}, \mathbf{o}^{(2)}, \dots, \mathbf{o}^{(M)}]$, and $\mathbf{o}^{(t)} = (\widetilde{\mathbf{W}})^T \widetilde{\mathbf{X}}^{(t)}$. The combination coefficient α can be solved efficiently by quadratic programming algorithms [7].

After obtaining the optimal \mathbf{Y} , we can get the optimal hashing code for the training set by thresholding \mathbf{Y} . More specifically, for \mathbf{Y}^p , which is the p -th row of \mathbf{Y} , if the j -th element \mathbf{Y}_j^p is larger than the specified threshold, $\mathbf{Y}_j^p = +1$, otherwise $\mathbf{Y}_j^p = -1$.

Then, a natural question would be: how to pick these thresholds? In [45], the authors pointed out that a good semantic hashing should also maximize the entropy to ensure efficiency. Using maximum entropy principle, a binary bit that gives balanced partitioning of the whole dataset always provides maximum information. Therefore, we set the threshold for binarising $\mathbf{Y}^p, p \in \{1, 2, \dots, K\}$ to be the median value of $\mathbf{v}_p, p \in \{1, 2, \dots, K\}$ (The median value for \mathbf{Y}^p will be denoted as \mathbf{m}_p thereafter). Still, after binarising, since different eigenvectors are mutually orthogonal, the different bits \mathbf{Y}^p will also be uncorrelated. In this way, the binary code achieves the best utilisation of the hash table and the constraint $\mathbf{Y}\mathbf{1} = 0$ in Eq.(9) can also be satisfied.

After getting the hashing code for the training set, the hashing code for the query example \mathbf{q} can be computed by first computing the hashing function $f(\mathbf{q}) = \sum_{t=1}^M \alpha_t (\widetilde{\mathbf{W}})^T \widetilde{\mathbf{q}}^{(t)}$, where, $\widetilde{\mathbf{q}}^{(t)}$ is derived from $\mathbf{q}^{(t)}$, using exactly the same way as we derive $\widetilde{\mathbf{x}}_i^{(t)}$ from $\mathbf{x}_i^{(t)}$ in Eq.(7). Then, the corresponding binary hashing code can be obtained through thresholding the hashing function output, i.e., the j -th hashing code for \mathbf{q} equals $+1$, if $f(\mathbf{q})_j > \mathbf{m}_j$ and otherwise -1 .

4.4 The Whole Algorithm

The training process of the proposed method is described in Table 1. The procedure of deriving hashing codes for query examples is summarized in Table 2. For hashing encoding, the training process is always conducted offline. Therefore, our focus of efficiency is on the prediction process. Since this process only involves some dot products

Algorithm: CHMIS-AW Predicting
Input: 1. A query example \mathbf{q} . 2. Hashing function weight vector $\widetilde{\mathbf{W}}$, 3. Convex combination coefficient α , 4. Median vector \mathbf{m} .
Output: The binary hashing code $\mathbf{h} \in \{-1, 1\}^{K \times 1}$ for \mathbf{q}
Initialization: 1. Construct $\widetilde{\mathbf{q}}^{(t)}$ similar to Eq.(7) For $t = 1:M$ 2. Calculate the regression output by $\mathbf{o}^{(t)} = (\widetilde{\mathbf{W}})^T \widetilde{\mathbf{q}}^{(t)}$ End For 3. Calculate the balanced output by $\mathbf{h} = f(\mathbf{q}) = \mathbf{O}\alpha$, where $\mathbf{O} = [\mathbf{o}^{(1)}, \mathbf{o}^{(2)}, \dots, \mathbf{o}^{(M)}]$ For $p = 1:K$ 4. if $\mathbf{h}_p > \mathbf{m}_p$, $\mathbf{h}_p = +1$, and otherwise $\mathbf{h}_p = -1$ End For

Table 2: Algorithm: CHMIS-AW Predicting

and aggregations between two vectors, which can be done in $O(s)$ time. Here, s is the average sparsity for features from the M sources.

4.5 Discussions

The CHMIS problem is related to the traditional multi-view learning [16, 21, 24, 40, 50, 53]. The basic motivation for many previous multi-view learning methods [16, 24, 40, 50] is to design a classifier on each view/source, and requires that the soft labels on these views/sources to be consistent and should not deviate too much. These methods are mainly designed for classification problems. However, in CHMIS, we need to learn two things from the training data, i.e., (1) the hashing codes for the training data, (2) a hashing function that enables efficient mappings of query examples. CHMIS is an unsupervised problem, in which we can consider the hashing codes as unknown labels. Therefore, the traditional multi-view learning methods cannot directly be applied to solve CHMIS.

In CHMIS-AW, we learn both the hashing codes and the hashing functions together in one formulation. They are both considered as optimization objectives in this formulation, so that the solution can be tuned to the best point between good hashing codes that preserves the similarities between examples in the original feature space and a perfect hashing function that approximates hashing codes for queries efficiently, since both of them are vital for efficient and effective retrieval. In contrast, in STH, the hashing codes and hashing functions are optimized in two different steps, which may cause a disconnection between them.

5. EXPERIMENTS

In this section, we will show the effectiveness of the proposed method through an extensive set of experiments².

5.1 Dataset

1. **Cora** [29] dataset consists of the abstracts and references of around 34,000 computer science papers. Part

²The code and data can be found on the author’s homepage.

of them are categorized into several subfields, such as Data Structure (DS), Hardware and Architecture (HA), Machine Learning (ML), Operation Systems (OS) and Programming Language (PL). We randomly choose OS among them to do comparison experiments. There are 4 topics in this dataset. The tf-idf (normalized term frequency and log inverse document frequency) features of the content part of these webpages are used as the first information source, and the link information is used as the second one. We randomly select 1122 examples as training examples, and another 124 example as testing examples/queries.

2. **Reuters (Reuters21578)**³ is a collection of documents that appeared on Reuters newswire in 1987. As the name suggests, it contains 21578 documents, with 135 categories. In our experiments, examples corresponding to the top 10 categories are kept, and the documents with more than one of the 10 categories are discarded. There are 7757 documents left. We use the original tf-idf content information, processed by PCA, as one information source, and the hidden topics information obtained from Probabilistic Latent Semantic Analysis (PLSA)⁴ of the binary word features as another one. 6982 examples are used as training examples, while 775 for testing.
3. **ReutersV1 (Reuters-Volume I)**: It is an archive of over 800,000 manually categorized newswire stories [23]. There are in total 126 topics in it. A subset of ReutersV1 is used. In experiments, we choose examples from ten categories of them. The documents with more than one category are discarded. There are in total 26161 examples left. 13081 examples are randomly selected as the training examples, while the remaining 13080 examples are used as testing examples. Similar to the Reuters (Reuters21578) dataset, we use the tf-idf features as one information source, and the dimensionality reduction results obtained from the PLSA of the binary word features as another information source.
4. **WebKB**⁵ consists of about 7000 webpages, collected from four universities, and is divided into 7 categories. We use content and link information as the two information sources of this dataset, where tf-idf features are used for the content part. 90% examples (6195) are randomly selected as training examples, while the remaining (688) examples are used for testing.
5. **Healthcare** dataset contains the information of 14199 diabetic patients over 1 year. The label of each patient indicates the patient has diabetes or not. We use the CCS hierarchy diagnosis code (where all the codes directly related to diabetic diagnosis have been discarded), CPT procedure code, and NDC drug code information as three different views of patient features

³<http://davidlewis.com/resources/textcollections/reuters21578/>.

⁴Actually, PLSA [20] can be considered as a dimensionality reduction method, which maps the documents into some fixed number of hidden topics. The topic distribution for each document can be used as low dimensional features.

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

	Cora			Reuters			ReutersV1			WebKB			Healthcare
	Combined	S1	S2	Combined									
CHMIS-AW	0.978	N/A	N/A	0.910	N/A	N/A	0.796	N/A	N/A	0.993	N/A	N/A	0.991
SH	0.957	0.514	0.329	0.902	0.883	0.820	0.778	0.752	0.718	0.612	0.779	0.584	0.877
STH	0.977	0.976	0.976	0.841	0.847	0.818	0.536	0.529	0.513	0.718	0.849	0.702	0.960
PCAH	0.959	0.861	0.634	0.881	0.862	0.793	0.701	0.674	0.642	0.652	0.809	0.595	0.801
LSH	0.961	0.901	0.718	0.892	0.846	0.820	0.790	0.677	0.620	0.867	0.853	0.647	0.832

Table 3: Precision for examples within Hamming distance 2, with 32 hashing bits. S1: Source 1, S2: Source2.

	Cora			Reuters			ReutersV1			WebKB			Healthcare
	Combined	S1	S2	Combined									
CHMIS-AW	0.495	N/A	N/A	0.831	N/A	N/A	0.717	N/A	N/A	0.478	N/A	N/A	0.795
SH	0.447	0.468	0.364	0.742	0.723	0.681	0.698	0.653	0.635	0.544	0.537	0.504	0.791
STH	0.468	0.475	0.461	0.784	0.751	0.720	0.554	0.530	0.523	0.407	0.456	0.404	0.766
PCAH	0.431	0.408	0.417	0.769	0.738	0.694	0.569	0.548	0.519	0.553	0.538	0.500	0.761
LSH	0.335	0.327	0.340	0.693	0.629	0.652	0.462	0.350	0.384	0.357	0.340	0.373	0.752

Table 4: Precision for the top 100 examples, with 32 hashing bits. S1: Source 1, S2: Source2.

to test our proposed algorithm. In particular, we used the information of 12780 patients as training set, while that of the remaining 1419 patients as testing.

5.2 Evaluation Metric

For each dataset, we use each document in the testing set as a query example to retrieve documents in the training set. We use two different metrics: the precision for the top 100 retrieved documents⁶, and the precision for the documents within the Hamming distance 2. Here, precision is defined as follows:

$$precision = \frac{\text{the number of retrieved relevant documents}}{\text{the number of all retrieved documents}}.$$

Throughout this paper, the relevant documents denote the ones with the same topics/labels as the query examples⁷. The averaged precision over all queries are recorded. As claimed in [22], once the number of bits is sufficiently high (e.g. 64), one would expect that distances with a Hamming distance less than or equal to two would correspond to nearest neighbors in the original data embedding.

For Reuters and healthcare dataset, we further report the precision-recall curves, where the recall rate is defined as:

$$recall = \frac{\text{the number of retrieved relevant documents}}{\text{the number of all relevant documents}}.$$

To draw the precision-recall curves, for each query example, we vary the number of retrieved documents from 0 to the number of all training examples, while fixing the number of hashing bits. The final results are the averaged results over all of the testing examples.

5.3 Comparison Methods

The proposed method CHMIS-AW is compared with four different algorithms on these datasets, i.e., Self Taught Hashing (STH), Spectral Hashing (SH), PCA Hashing (PCAH), and Latent Semantic Hashing (LSH).

These four comparison methods cannot be directly used for the multiple information sources. So, when conducting these comparison experiments, we first concatenate the features on different sources, use them as a single set of features and then apply the comparison algorithms. The parameters C_1 and C_2 in CHMIS-AW are tuned by 5-fold cross validation on the training set through the grid $\{0.5, 2, 4, 8, 16, 32, 128\}$.

⁶If there are ties in the desired Hamming distance, some examples are randomly picked.

⁷Please note that the labels are only used for evaluation purpose, but not for training.

When constructing the graph laplacians for both CHMIS-AW and STH, the number of nearest neighbors is fixed to be 7 for all experiments. For LSH, we randomly select projections from a Gaussian distribution with zero-mean and identity covariance to construct the hash tables.

5.4 Results and Discussions

First of all, we evaluate the performances of different algorithms by varying the number of hashing bits in $\{8, 16, 32, 64, 128\}$. The precision for the top 100 retrieved examples with different number of hashing bits is reported in Fig.1. And the precision for examples within hamming distance 2 is reported in Fig.2. From these comparison results, we can see CHMIS-AW shows the best performance among five hashing methods in most of the cases with information from different sources. CHMIS-AW also outperforms the other hashing methods in most cases when they use information from one specific source. In Fig.1(d), the precision of the proposed method is lower than that of SH and PCAH. However, from the perspective of precision within hamming distance 2, as shown in Fig.2(d), it gives the best performance. So, we can conclude that, in WebKB, CHMIS-AW achieves the best performance in retrieving the several most relevant examples, although probably not the top 100. The good performance of the CHMIS-AW is mainly because: (1) CHMIS-AW integrates the different information sources together more naturally; (2) CHMIS-AW learns the hashing codes and hashing functions in one formulation, which effectively avoid the noise that can be caused in a two step’s method as in STH.

As claimed in [33, 47], the LSH method is data-oblivious, and may lead to inefficient codes in practice. From the reported results, we can see that, LSH does not perform well in most cases, especially under the criteria of the precision of the top 100 examples. But for the criteria of the precision within Hamming distance 2, LSH does well in some cases. The similar behavior is also observed in [22]. As a spectral based method, SH borrows the idea of spectral analysis. Its basic motivation is reasonable. However, in CHMIS, SH cannot find a good similarity matrix that can be consistent on the different sources. As shown in the experimental results, the performance of SH is worse than that of CHMIS-AW. STH is a two step method. The first step is spectral clustering step, and the second step employs SVM. However, it still cannot integrate the natural of multiple source problem into the algorithm and it needs to prepare the training data in a format for SVM, which may introduce some noise in this

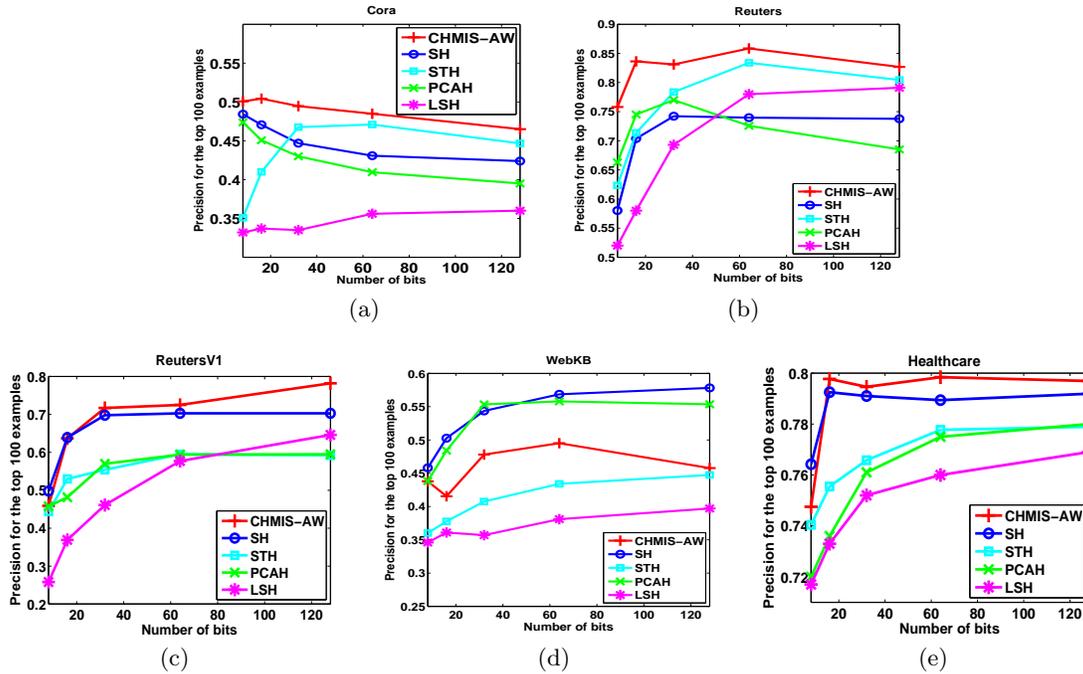


Figure 1: Precision results for the top 100 returned results. It is clear that CHMIS-AW shows the best performance among the five hashing methods in most of the cases.

process. That’s why its performance cannot exceed that of CHMIS-AW.

By fixing the number of bits to 32, we further report the corresponding performances on all of the five datasets, along with the performance on each individual source in Tables 3 and 4. It seems that, in some cases, brute force concatenating the features from different sources together may not necessarily improve the hashing performance. It further confirms that we need some more sophisticated algorithms specifically designed for CHMIS. Since the proposed method is specifically for CHMIS, it achieves the best performance in most of scenarios under both of the two criteria. The precision-recall curves with 32 hashing bits on Reuters and healthcare datasets are reported in Fig.3(a) and Fig.3(b) respectively. It is clear that among all of these comparison methods, CHMIS-AW shows the best performance. STH performs better than SH, PCAH and LSH. We have also observed similar results on the other three datasets. But due to the limit of space, they cannot be reported here.

There are two parameters in the proposed method, i.e., C_1 and C_2 . To prove the robustness of the proposed method, we conduct some parameter sensitivity experiments on Reuters. For each experiment, we tune only one parameter from the grid $\{0.5, 2, 4, 8, 16, 32, 128\}$, while holding the other one. The results are reported in Fig. 4 and Fig. 5. It is clear from these experimental results that the performance of the proposed method are relatively stable with respect to C_1 and C_2 . We have also observed similar patterns of the proposed method in the other four datasets.

The testing process of CHMIS-AW is fast, since the hashing function is a linear mapping and only involves some dot products. On an ordinary PC with Intel Core Duo CPU 2.5 GHZ and 4GB RAM, it takes about 0.0001 second per ex-

ample for prediction in all datasets, which is similar to the comparison methods.

6. FUTURE WORKS

In this paper, in Eq.(3), we combined the graph Laplacian on each individual source with equal weights. However, a natural question would be: since we are tuning the importance on each individual source by using a balancing factor α , can we do the same thing to combine the different graph Laplacian with different weights? One choice is to change the objective function in Eq.(9) to:

$$tr(C_1 \mathbf{Y}^T \sum_{t=1}^M g(\alpha_t) \tilde{\mathbf{L}}^{(t)} \mathbf{Y}) + C_2 \|\mathbf{Y} - \sum_{t=1}^M \alpha_t (\tilde{\mathbf{W}})^T \tilde{\mathbf{X}}^{(t)}\|^2 + \|\tilde{\mathbf{W}}\|^2,$$

where $g(\bullet) : \mathbf{R} \rightarrow \mathbf{R}$ is a non-decreasing function. We believe this is true. The problem is how to design a reasonable $g(\bullet)$ function for this objective function. It is an intriguing problem that needs to be solved in the future.

Besides the problem of combining graph Laplacians, another future direction is to investigate how to solve the problem when some sources of features are missing for some specific examples.

7. CONCLUSIONS

To enable fast similarity retrieval, semantic hashing methods represent the examples by a small number of binary bits, so that the storage space can be minimized, and the retrieval speed can be accelerated. However, previous hashing methods only consider the case when each example is represented by one type of features. There is no prior work that can incorporate different information sources together and learn the corresponding hashing codes. To address this problem,

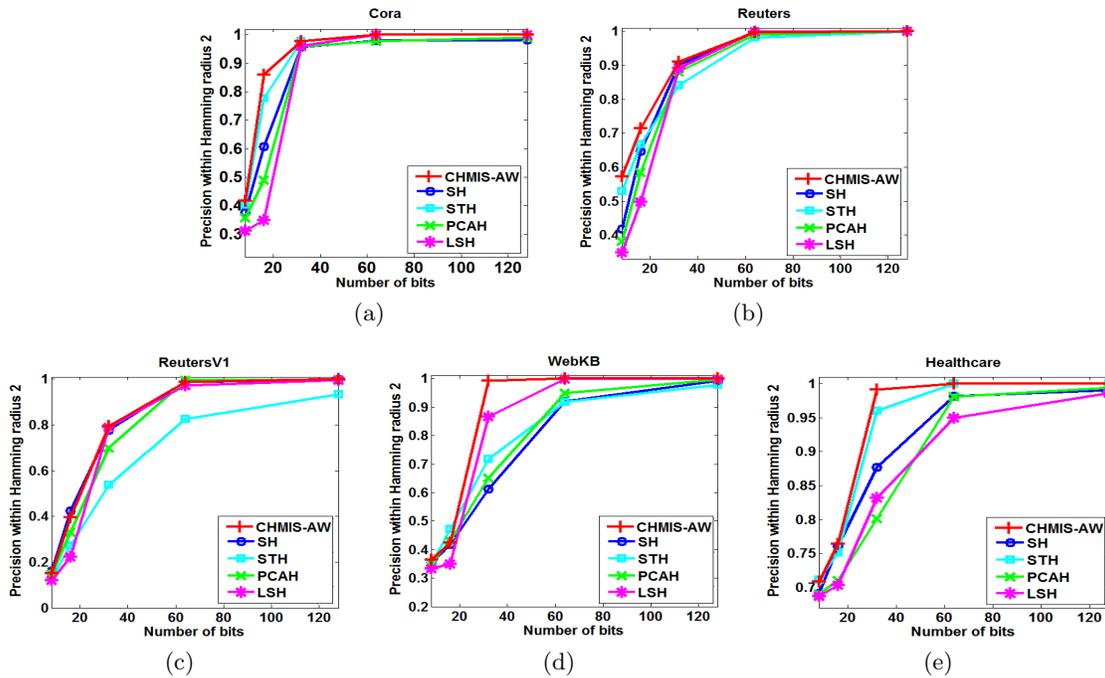


Figure 2: Precision results for the top results within hamming distance 2. It is clear that CHMIS-AW shows the best performance among the five hashing methods in most of the cases.

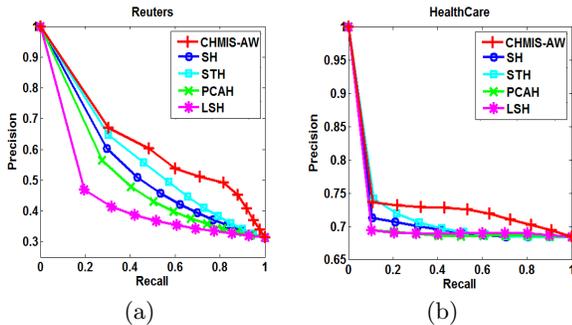


Figure 3: Precision and Recall Curve on Reuters and the healthcare dataset, with fixed bit number 32. CHMIS-AW demonstrates its superior performance over the other three algorithms.

this paper proposes a novel research problem as Composite Hashing with Multiple Information Sources (CHMIS) for intelligently combining information from different sources into final hashing codes. In particular, a method called CHMIS-AW (Adjusted Weights) is proposed to achieve this goal. CHMIS-AW is an iterative method, which optimizes the relaxed hashing codes and the combination coefficients alternatively. An extensive set of experiments clearly demonstrates the superior performance of the proposed method against several state-of-the-art techniques.

8. ACKNOWLEDGEMENT

We thank the anonymous reviewers for valuable comments. This research was partially supported by the NSF research grants IIS-0746830, CNS-1012208, IIS-1017837, CCF- 0939370.

References

- [1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *FOCS*, pages 459–468, 2006.
- [2] R. Baeza-Yates, C. Castillo, F. Junqueira, V. Plachouras, and F. Silvestri. Challenges in distributed information retrieval. In *ICDE*, 2007.
- [3] R. Baeza-Yates, B. Ribeiro-Neto, et al. *Modern information retrieval*, volume 463. ACM press New York, 1999.
- [4] S. Baluja and M. Covell. Learning to hash: forgiving hash functions and applications. *Data Mining and Knowledge Discovery*, 17(3):402–430, 2008.
- [5] Y. Bengio, O. Delalleau, N. Roux, J. Paiement, P. Vincent, and M. Ouimet. Learning eigenfunctions links spectral embedding and kernel PCA. *Neural Computation*, 16(10):2197–2219, 2004.
- [6] C. Bishop et al. *Pattern recognition and machine learning*. Springer New York, 2006.
- [7] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge Univ Pr, 2004.
- [8] A. Z. Broder. On the resemblance and containment of documents. In *CCS*, pages 21–29. IEEE Computer Society, 1997.
- [9] O. Chum, M. Perdoch, and J. Matas. Geometric min-hashing: Finding a (thick) needle in a haystack. In *CVPR*, pages 17–24, 2009.
- [10] O. Chum, J. Philbin, and A. Zisserman. Near duplicate image detection: min-hash and tf-idf weighting. In *BMVC*, volume 3, page 4, 2008.
- [11] F. Chung and N. Biggs. Spectral graph theory. *American Mathematical Society Providence, RI*, 1997.
- [12] T. Cormen, C. Leiserson, R. Rivest, and C. Stein. *Introduction to algorithms*. The MIT press, 2001.
- [13] J. S. Culpepper and A. Moffat. Efficient set intersection for inverted indexing. *ACM Trans. Inf. Syst.*, 29:111–125, December 2010.
- [14] M. Datar, N. Immorlica, P. Indyk, and V. S. Mirrokni. Locality-sensitive hashing scheme based on p-stable distributions. In *SCG*, pages 253–262, 2004.
- [15] 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.

[16] J. Farquhar, D. Hardoon, H. Meng, J. Shawe-Taylor, and S. Szepesvári. Two view learning: SVM-2K, theory and practice. *NIPS*, 18:355, 2006.

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

[18] G. E. Hinton, S. Osindero, and Y.-W. Teh. A Fast Learning Algorithm for Deep Belief Nets. *Neural Comp.*, 18(7):1527–1554, July 2006.

[19] G. E. Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks. *Science*, 313(5786):504–507, July 2006.

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

[21] T. Joachims and N. Cristianini. Composite kernels for hypertext categorisation. In *ICML*, pages 250–257. Morgan Kaufmann Publishers, 2001.

[22] B. Kulis and T. Darrell. Learning to hash with binary reconstructive embeddings. *NIPS*, 2009.

[23] 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.

[24] G. Li, S. C. H. Hoi, and K. Chang. Two-view transductive support vector machines. In *SDM*, pages 235–244, 2010.

[25] J. Lin. Brute force and indexed approaches to pairwise document similarity comparisons with mapreduce. In *SIGIR*, pages 155–162. ACM, 2009.

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

[27] D. Lowe. Object recognition from local scale-invariant features. In *ICCV*, page 1150, 1999.

[28] C. Manning, P. Raghavan, and H. Schütze. An introduction to information retrieval. *Cambridge University Press*, 2008.

[29] A. K. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3:127–163, 1999.

[30] Y. Mu, J. Shen, and S. Yan. Weakly-supervised hashing in kernel space. In *CVPR*, pages 3344–3351, 2010.

[31] R.E.Schapire. The boosting approach to machine learning: An overview. *Nonlinear Estimation and Classification*, pages 149–172, 2003.

[32] D. Rosenberg, V. Sindhwani, P. Bartlett, and P. Niyogi. A Kernel for Semi-Supervised Learning With Multi-View Point Cloud Regularization. *IEEE Signal Processing Magazine*, 2009.

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

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

[35] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information processing and management*, 24(5):513–523, 1988.

[36] B. Scholkopf and A. Smola. *Learning with kernels*. MIT press Cambridge, Mass, 2002.

[37] G. Shakhnarovich, P. A. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *ICCV*, pages 750–759, 2003.

[38] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 2002.

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

[40] V. Sindhwani and P. Niyogi. A co-regularized approach to semi-supervised learning with multiple views. In *ICML Workshop on Learning with Multiple Views*, 2005.

[41] B. Stein. Principles of hash-based text retrieval. In *SIGIR*, page 534. ACM, 2007.

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

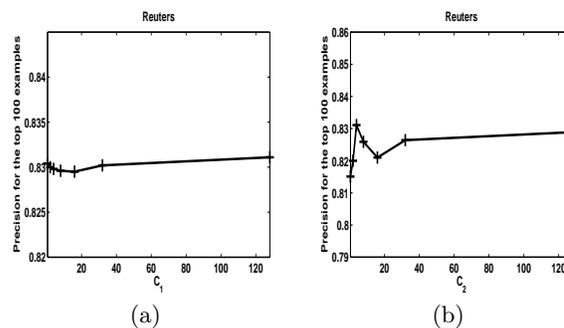


Figure 4: Parameter Sensitivity for C_1 and C_2 , with 32 hashing bits. It shows that the proposed method is relatively stable with the two parameters for the precision of the top 100 retrieved documents.

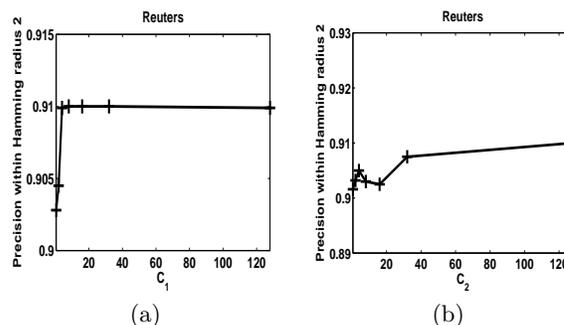


Figure 5: Parameter Sensitivity for C_1 and C_2 , with 32 hashing bits. The proposed method is stable with the two parameters under the precision within hamming distance 2.

[43] J. Wang, O. Kumar, and S.-F. Chang. Semi-supervised hashing for scalable image retrieval. In *CVPR*, pages 3424–3431, 2010.

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

[45] X. Wang, L. Zhang, F. Jing, and W. Ma. Annosearch: Image auto-annotation by search. In *CVPR*, 2006.

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

[47] Y. Weiss, A. Torralba, and R. Fergus. Spectral hashing. *NIPS*, 21:1753–1760, 2009.

[48] I. H. Witten, I. H. Witten, A. Moffat, and T. C. Bell. *Managing Gigabytes: Compressing and Indexing Documents and Images*. Morgan Kaufmann, May 1999.

[49] L. Zelnik-Manor and P. Perona. Self-tuning spectral clustering. *NIPS*, 17:1601–1608, 2004.

[50] D. Zhang, F. Wang, C. Zhang, and T. Li. Multi-view local learning. In *AAAI*, pages 752–757, 2008.

[51] D. Zhang, J. Wang, D. Cai, and J. Lu. Laplacian co-hashing of terms and documents. *Advances in Information Retrieval*, pages 577–580, 2010.

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

[53] T. Zhang, A. Popescul, and B. Dom. Linear prediction models with graph regularization for web-page categorization. In *SIGKDD*, pages 821–826. ACM, 2006.

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