Sparse Semantic Hashing for Efficient Large Scale Similarity Search

Qifan Wang, Bin Shen, Zhiwei Zhang and Luo Si
Department of Computer Science
Purdue University
West Lafayette, IN 47907, US
{wang868, bshen, zhan1187, lsi}@purdue.edu

ABSTRACT

Similarity search, or finding approximate nearest neighbors, is an important technique in various large scale information retrieval applications such as document retrieval. Many recent research demonstrate that hashing methods can achieve promising results for large scale similarity search due to its computational and memory efficiency. However, most existing hashing methods ignore the hidden semantic structure of documents but only use the keyword features (e.g., tf-idf) in hashing codes learning. This paper proposes a novel sparse semantic hashing (SpSH) approach that explores the hidden semantic representation of documents in learning their corresponding hashing codes. In particular, a unified framework is designed for ensuring the hidden semantic structure among the documents by a sparse coding model, while at the same time preserving the document similarity via graph Laplacian. An iterative coordinate descent procedure is then proposed for solving the optimization problem. Extensive experiments on two large scale datasets demonstrate the superior performance of the proposed research over several state-of-the-art hashing methods.

Categories and Subject Descriptors

H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing

Keywords

Hashing; Similarity Search; Sparse Coding

1. INTRODUCTION

Similarity search, also known as approximate nearest neighbor search, is a key problem in many information retrieval applications including document and image retrieval [5], similar content reuse detection [11] and collaborative filtering [12]. The purpose of similarity search is to identify similar data examples given a query example. With the explosive growth of the internet, a huge amount of data have been generated, which indicates that efficient similarity search with large scale data becomes more important. Traditional similarity search methods are difficult to be used directly for large scale data since computing the similarity using the original features (i.e., often in high dimensional space) exhaustively between the query example and every candidate example is impractical for large applications. Recently, hashing methods [6, 7, 8, 9, 10] have been successfully used for large scale similarity search due to its fast query speed and low storage cost. These hashing methods design compact binary code in a low-dimensional space for each document so that similar documents are mapped to similar binary codes. In the retrieval process, these hashing methods first transform each query example into its corresponding binary code. Then similarity search can be simply conducted by calculating the Hamming distances between the codes of available data examples and the query and selecting data examples within small Hamming distances, which can be calculated using efficient bitwise operator XOR.

Locality-Sensitive Hashing (LSH) [1] is one of the most commonly used data-independent hashing methods. It utilizes random linear projections, which are independent of training data, to map data points from a high-dimensional feature space to a low-dimensional binary space. Another class of hashing methods are called data-dependent methods, whose projection functions are learned from training data. These data-dependent methods include spectral hashing (SH) [9], principal component analysis based hashing (PCAH) [4], self-taught hashing (STH) [10] and iterative quantization (ITQ) [3]. SH learns the hashing codes based on spectral graph partitioning and forcing the balanced and uncorrelated constraints into the learned codes. PCAH utilizes principal component analysis (PCA) to learn the projection functions. STH combines an unsupervised learning step with a supervised learning step to learn effective hashing codes. ITQ learns an orthogonal rotation matrix to refine the initial projection matrix learned by PCA so that the quantization error of mapping the data to binary codes is minimized. Compared with the data-independent methods, these data-dependent methods generally provide more effective hashing codes.

Hashing methods generate promising results by successfully addressing the storage and search efficiency challenges. However, most existing hashing methods ignore the hidden semantic structure of documents but only learns hashing codes using original features (i.e., tf-idf). In document
retrieval, the hidden semantics usually reflect the true meanings/categories of a document. It is more desirable to find those documents that share same semantics instead of keywords to a query. In other words, the semantic captures the hidden information contained in a document and thus can better represent documents than original keyword features. Therefore, it is important to design hashing method that preserve the semantic structure among documents in the learned Hamming space.

This paper proposes a novel Sparse Semantic Hashing (SpSH) approach that explores the hidden semantic representation of documents in learning their corresponding hashing codes. In particular, a unified framework is designed for ensuring the hidden semantic structure among the documents by a sparse coding model, while at the same time preserving the document similarity using graph Laplacian. An iterative coordinate descent procedure is then proposed for solving the optimization problem. Extensive experiments on two large scale datasets demonstrate the superior performance of the proposed research over several state-of-the-art hashing methods.

2. SPARSE SEMANTIC HASHING

This section first states the problem setting of SpSH. Assume there are total \( n \) training data examples, denoted as: 
\[
X = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^{d \times n},
\]
where \( d \) is the dimensionality of the feature. The main purpose of SpSH is to map these training examples to the optimal binary hashing codes 
\[
Y = \{y_1, y_2, \ldots, y_n\} \in \{0, 1\}^{k \times n}
\]
through a hashing function 
\[
f : \mathbb{R}^d \rightarrow \{0, 1\}^k
\]
such that the similarities among data examples in original feature space are preserved in the hashing codes. Here \( k \) is the number of hashing bits and 
\[
y_j = f(x_j)
\]

2.1 Problem Formulation

The proposed SpSH 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 learning the hidden semantic representation of documents and preserving the document similarity. In particular, the objective function of SpSH is composed of two components: (1) Semantic representation component, which ensures that the hashing codes are consistent with hidden semantics via a sparse coding model [2]; (2) Similarity preservation component, which aims at preserving the document 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.

2.1.1 Sparse Semantic Reconstruction

The goal of semantic reconstruction of documents is to learn a basis \( B \in \mathbb{R}^{d \times k} \) and corresponding sparse codes such that input data can be well approximated/represented. Here we assume there are \( k \) hidden semantics, each represented by a column of basis \( B \). For the \( k \) hashing bits of each document, if the document contains the \( j \)-th semantic, then its corresponding \( j \)-th bit should be 1, otherwise 0. In this way, the hashing code essentially represents the hidden semantics of the document. Since a document usually related to a small number of semantics, we impose a sparse constraint to ensure that there are few 1’s in the hashing code. Then the sparse semantic reconstruction term can be written as:

\[
||X - BY||_F^2 + \alpha||B||_F^2 + \gamma||Y||_1
\]  

(1)

here \( ||.||_F \) is the matrix Frobenius norm. \( ||B||_F^2 \) is introduced to avoid overfitting. \( ||Y||_1 \) is the sparsity constraint. \( \alpha \) and \( \gamma \) are the weight parameters. Intuitively, we reconstruct each document in the corpus \( X \) using a small number of basis in \( B \) indicated by the hashing code, where a 1 in the code means the corresponding semantic is related to the document and a 0 means irrelevant semantic. By minimizing this term, the hidden semantic structure among the documents are preserved in the learned hashing codes.

2.1.2 Similarity Preservation

One of the key problems in hashing algorithms is similarity preserving, which indicates that similar documents should be mapped to similar hashing codes within a short Hamming distance. The Hamming distance between two binary codes \( y_i \) and \( y_j \) can be calculated as \( \frac{1}{2}||y_i - y_j||^2 \). To measure the similarity between documents represented by the binary hashing codes, one natural way is to minimize the weighted average Hamming distance as follows:

\[
\sum_{i,j} S_{ij} ||y_i - y_j||^2
\]  

(2)

Here, \( S \) is the similarity matrix which is calculated based on the document features. To meet the similarity preservation criterion, we seek to minimize this quantity, because it incurs a heavy penalty if two similar documents are mapped far away. There are many different ways to define the similarity matrix \( S \). In this paper, we adopt the local similarity due to its nice property in many information retrieval applications [7, 10]. In particular, the corresponding similarities are computed by Gaussian functions, i.e.,
\[
S_{ij} = e^{-||x_i - x_j||^2/s^2_i},
\]
where \( s_{ij} \) is a scaling parameter.

By introducing a diagonal \( n \times n \) matrix \( D \), whose entries are given by \( D_{ii} = \sum_{j=1}^{n} S_{ij} \), Eqn.1 can be rewritten as:

\[
tr(Y(D - S)Y^T) = tr(YLY^T)
\]  

(3)

where \( L \) is the graph Laplacian and \( tr() \) is the matrix trace. By minimizing this term, the similarity between different documents can be preserved in the learned hashing codes.

2.2 Optimization Algorithm

The entire objective function of the proposed SpSH combines the above two components as follows:

\[
\min_{B,Y} ||X - BY||_F^2 + \alpha||B||_F^2 + \beta tr(YLY^T) + \gamma||Y||_1
\]  

s.t. \( Y \in \{0, 1\}^{k \times n} \)

(4)

2.2.1 Relaxation

Directly minimizing the objective function in Eqn.4 is intractable because of the discrete constraint. Therefore, we propose to relax this constraint to \( 0 \leq Y \leq 1 \). However, even after the relaxation, the objective function is still difficult to optimize since \( Y \) and \( B \) are coupled together and it is non-convex with respect to \( Y \) and \( B \) jointly. We propose to use a coordinate descent algorithm for solving this relaxed optimization problem by iteratively optimizing the objective with respect to \( Y \) and \( B \). In particular, after initializing \( B \),
the relaxed problem can be solved by doing the following two steps iteratively until convergence.

**Step 1:** Fix $B$, optimize w.r.t. $Y$:

$$
\min_Y \|X - BY\|_F^2 + \beta \text{tr}(LYY^T) + \gamma \|Y\|_1 \tag{5}
$$

The objective function is differentiable with respect to $Y$ and the partial derivative of Eqn.5 can be calculated as:

$$
\frac{\partial \text{Eqn}(5)}{\partial Y} = -2B^T X + 2B^T BY + 2\beta Y + \gamma I \tag{6}
$$

With this obtained gradient, L-BFGS Quasi-Newton method is applied to solve this optimization problem.

**Step 2:** Fix $Y$, solve for $B$:

$$
\min_B \|X - BY\|_F^2 + \alpha \|B\|_F^2 \tag{7}
$$

We can obtain the close form solution of $B$ as:

$$
B = XY^T (YY^T + \alpha I)^{-1} \tag{8}
$$

By solving Eqns.5 and 7 iteratively, the optimal values of $Y$ and $B$ can be obtained.

### 2.2.2 Binarization

After obtaining the optimal solution for the relaxed problem, we need to binarize them to obtain binary hashing codes that satisfy the relaxed constraints. The binary hashing codes for the training set can be obtained by thresholding $Y$. It was pointed out in [4] and [7] that desired hashing codes should also maximize the entropy to ensure efficiency. Following the maximum entropy principle, a binary bit that gives balanced partitioning of the whole dataset should provide maximum information. Therefore, we set the threshold for binarizing the $p$-th bit to be the median of $y_p$. In particular, if $p$-th bit of $y_j$ is larger than median value, $y_j^p$ is set to 1, otherwise $y_j^p$ is set to 0. In this way, the binary code achieves the best balance.

### 2.2.3 Hashing Function

A linear hashing function is utilized to map documents to the binary hashing codes as:

$$
y_j = f(x_j) = H x_j \tag{9}
$$

where $H$ is a $k \times d$ parameter matrix representing the hashing function. Then the optimal hashing function can be obtained by minimizing $\|Y - HX\|^2$.

## 3. EXPERIMENTS

### 3.1 Datasets and Implementation

Two text datasets are used in our experiments. **ReutersV1** dataset contains over 800,000 manually categorized newswire stories. 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. **20Newsgroups** corpus is collected and originally used for document categorization. We use the popular ‘18828’ version which contains 18828 documents. 16946 documents are randomly chosen for training and the rest 1882

![Figure 1](http://www.daviddowe.com/resources/text/rcv1/2http://people.csail.mit.edu/jennie/20Newsgroups/)## 3.2 Evaluation Method

The search results are evaluated based on the ground-truth labels. We use several metrics to measure the performance of different methods. For evaluation with **Hamming Ranking**, we calculate the precision at top $k$ that is the percentage of relevant neighbors among the top $k$ returned examples, where we set $k$ to be 100 in the experiments. For evaluation with **Hash Lookup**, all the examples within a fixed Hamming distance, $r$, of the query are evaluated. In particular, following [5] and [9], a Hamming distance $r = 2$ is used to retrieve the neighbors in the case of **Hash Lookup**. The precision of the returned examples falling within Hamming distance 2 is reported.

### 3.3 Results and Discussion

The proposed SpSH approach is compared with five different methods, i.e., Spectral Hashing (SH) [9], PCA Hashing (PCAH) [4], Latent Semantic Hashing (LSH) [1], Self Taught Hashing (STH) [10] and Iterative Quantization (ITQ) [3]. We evaluate the performance of different methods by varying the number of hashing bits in the range of \{8, 16, 32, 64, 128\}.

Three sets of experiments are conducted on both datasets to evaluate the performance of SpSH. In the first set of experiments, we report the precision values for the top 100
Figure 2: Precision-Recall behavior on two datasets.

retrieved examples in Fig.1(a)-(b). The precision values for retrieved examples with Hamming distance 2 are reported in Fig.1(c)-(d). From these comparison results, we can see that SpSH achieves the best performance among all compared hashing methods on both datasets. We also observe from Fig.1(c)-(d) that the precisions of Hash Lookup decrease significantly with the increasing number of hashing bits. This is because when using longer hashing bits, the Hamming space becomes increasingly sparse and very few data points fall within the Hamming ball with radius 2, resulting in many 0 precision queries. However, the precision values of SpSH are still consistently higher than other methods.

In the second set of experiments, the precision-recall curves with 32 hashing bits on both datasets are reported in Fig.2. It is clear that among all of these comparison methods, SpSH shows the best performance. From the reported figures, we can see that LSH does not perform well in most cases. This is because the LSH method is data-independent and may lead to inefficient codes in practice. For SH and STH, although these methods try to preserve the similarity between data examples in their learned hashing codes, they do not model the hidden semantic structure among the documents while our SpSH learns a better sparse semantic representations for document corpus. ITQ achieves better performance than SH and STH since it somehow tries to minimize the quantization errors. However, different from these methods, the proposed SpSH learns the optimal hashing codes and hidden semantic basis jointly to better represent the documents and thus achieves better hashing performance.

The third set of experiments study the training cost for learning hashing function and testing cost for encoding each query. The results on both datasets with 32 bits is reported in Table 1. We can see from this table that the training cost of SpSH is around one hundred seconds, which is comparable with most of the other hashing methods and it is not slow in practice considering the complexity of training. The test time for SpSH is sufficiently fast especially when compared to the nonlinear hashing method SH. The reason is that it only needs linear projection and binarization to generate the hashing codes for queries.

4. CONCLUSION

This paper proposes a novel sparse semantic hashing approach that explores the hidden semantic representation of documents in learning their corresponding hashing codes.

<table>
<thead>
<tr>
<th>Methods</th>
<th>ReutersV1</th>
<th></th>
<th>20Newsgroups</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>SpSH</td>
<td>123.86</td>
<td>0.4x10^{-4}</td>
<td>74.84</td>
<td>0.5x10^{-4}</td>
</tr>
<tr>
<td>ITQ</td>
<td>45.34</td>
<td>0.4x10^{-4}</td>
<td>11.19</td>
<td>0.5x10^{-4}</td>
</tr>
<tr>
<td>STH</td>
<td>76.39</td>
<td>0.4x10^{-4}</td>
<td>48.77</td>
<td>0.5x10^{-4}</td>
</tr>
<tr>
<td>SH</td>
<td>58.37</td>
<td>3.6x10^{-4}</td>
<td>16.23</td>
<td>3.9x10^{-4}</td>
</tr>
<tr>
<td>PCAH</td>
<td>23.17</td>
<td>0.4x10^{-4}</td>
<td>8.18</td>
<td>0.5x10^{-4}</td>
</tr>
<tr>
<td>LSH</td>
<td>2.24</td>
<td>0.4x10^{-4}</td>
<td>2.21</td>
<td>0.5x10^{-4}</td>
</tr>
</tbody>
</table>

Table 1: Training and testing time (in second) on two datasets with 32 hashing bits.

A unified framework is designed for ensuring the hidden semantic structure among the documents by a sparse coding model, while at the same time preserving the document similarity using graph Laplacian. Extensive experiments on two datasets demonstrate the superior performance of the proposed research over several state-of-the-art hashing methods.

5. ACKNOWLEDGMENTS

This work is partially supported by NSF research grants IIS-0746830, DRL-0822926, CNS-1012208, IIS-1017837, CNS-1314688 and a research grant from Office of Naval Research (ONR-11627465). 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.

6. REFERENCES