

Extensions to Self-Taught Hashing: Kernelisation and Supervision

Dell Zhang
DCSIS
Birkbeck, University of London
Malet Street
London WC1E 7HX, UK
dell.z@ieee.org

Deng Cai
State Key Lab of CAD&CG
Zhejiang University
100 Zijingang Road
Hangzhou 310058, China
dengcai@cad.zju.edu.cn

Jun Wang
Dept of Computer Science
University College London
Gower Street
London WC1E 6BT, UK
jun.wang@cs.ucl.ac.uk

Jingsong Lu
DEMS
Birkbeck, University of London
Malet Street
London WC1E 7HX, UK
jingsong.lu@gmail.com

ABSTRACT

The ability of fast similarity search at large scale is of great importance to many Information Retrieval (IR) applications. A promising way to accelerate similarity search is semantic hashing which designs compact binary codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance). Since each bit in the binary code for a document can be regarded as a binary feature of it, semantic hashing is essentially a process of generating a few most informative binary features to represent the documents. Recently, we have proposed a novel Self-Taught Hashing (STH) approach to semantic hashing (that is going to be published in SIGIR-2010): we first find the optimal l -bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l -bit code for any query document unseen before. In this paper, we present two further extensions to our STH technique: one is kernelisation (i.e., employing nonlinear kernels to achieve nonlinear hashing), and the other is supervision (i.e., exploiting the category label information to enhance the effectiveness of hashing). The advantages of these extensions have been shown through experiments on synthetic datasets and real-world datasets respectively.

Categories and Subject Descriptors

H.2.8 [Database Management]: Database Applications—*data mining*; H.3.1 [Information Storage and Retrieval]: Content Analysis and Indexing; H.3.3 [Information Storage and Retrieval]: Information Search and Retrieval;

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'10, July 19–23, 2010, Geneva, Switzerland.

Copyright 2010 ACM 978-1-60558-896-4/10/07 ...\$10.00.

I.2.6 [Artificial Intelligence]: Learning; I.5.2 [Pattern Recognition]: Design Methodology—*classifier design and evaluation*

General Terms

Algorithms, Experimentation, Performance

Keywords

Similarity Search, Semantic Hashing, Laplacian Eigenmap, Support Vector Machine, Feature Generation.

1. INTRODUCTION

The problem of *similarity search* (aka *nearest neighbour search*) is: given a query document, find its most similar documents from a very large document collection (corpus). It is of great importance to many Information Retrieval (IR) [26] applications, such as near-duplicate detection [18], plagiarism analysis [35], collaborative filtering [23], caching [28], and content-based multimedia retrieval [25].

Recently, with the rapid evolution of the Internet and the increased amounts of data to be processed, how to conduct fast similarity search at large scale has become an urgent research issue. A promising way to accelerate similarity search is *semantic hashing* [29] which designs compact binary codes for a large number of documents so that semantically similar documents are mapped to similar codes (within a short Hamming distance). It is extremely fast to perform similarity search over such binary codes, because we can simply return all the documents that are hashed into a tight Hamming ball centred around the binary code of the query document [34].

Since each bit in the binary code for a document can be regarded as a binary feature of it, semantic hashing is essentially a process of generating a few most informative binary features to represent the documents.

In our paper that is going to appear in the Proceedings of SIGIR-2010 [41], we have proposed a novel Self-Taught Hashing (STH) approach to semantic hashing: we first find the optimal l -bit binary codes for all documents in the given

corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l -bit code for any query document unseen before. STH using binarised Laplacian Eigenmap (LapEig) [3] and linear Support Vector Machine (SVM) [22, 31] significantly outperforms state-of-the-art techniques in our experiments.

In this paper that has been accepted by the SIGIR-2010 Workshop on Feature Generation and Selection for Information Retrieval (FGSIR), we present two further extensions to our STH technique: one is kernelisation (i.e., employing nonlinear kernels to achieve nonlinear hashing), and the other is supervision (i.e., exploiting the category label information to enhance the effectiveness of hashing). The advantages of these extensions have been shown through our experiments on synthetic datasets and real-world datasets respectively.

The rest of this paper is organised as follows. In Section 2, we describe the related work. In Section 3, we review our recently proposed STH technique. In Section 4, we present two further extensions to STH, kernelisation and supervision, in details. In Section 5, we make conclusions.

2. RELATED WORK

There has been extensive research on fast similarity search due to its central importance in many applications. For a low-dimensional feature space, similarity search can be carried out efficiently with pre-built space-partitioning index structures (such as KD-tree) or data-partitioning index structures (such as R-tree) [8]. However, when the dimensionality of feature space is high (say > 10), similarity search aiming to return exact results cannot be done better than the naive method — a linear scan of the entire collection [37]. In the IR domain, documents are typically represented as feature vectors in a space of more than thousands of dimensions [26]. Nevertheless, if the complete exactness of results is not really necessary, similarity search in a high-dimensional space can be dramatically speeded up by using hash-based methods which are purposefully designed to approximately answer queries in virtually constant time [34].

Such hash-based methods for fast similarity search can be considered as a means for embedding high-dimensional feature vectors to a low-dimensional Hamming space (the set of all 2^l binary strings of length l), while retaining as much as possible the semantic similarity structure of data. Unlike standard dimensionality reduction techniques such as Latent Semantic Indexing (LSI) [5, 10] and Locality-Preserving Indexing (LPI) [17, 16], hashing techniques map feature vectors to *binary* codes, which is key to extremely fast similarity search (see Section 1). One possible way to get binary codes for text documents is to binarise the real-valued low-dimensional vectors (obtained from dimensionality reduction techniques like LSI) via thresholding [29]. An improvement on binarised-LSI that directly optimises a Hamming distance based objective function, namely Laplacian Co-Hashing (LCH), has been proposed recently [40].

The most well-known hashing technique that preserves similarity information is probably Locality-Sensitive Hashing (LSH) [1]. LSH simply employs random linear projections (followed by random thresholding) to map data points close in a Euclidean space to similar codes. It is theoretically guaranteed that as the code length increases, the Hamming distance between two codes will asymptotically approach the Euclidean distance between their corresponding data points. However, since the design of hash functions for LSH is *data-*

oblivious, LSH may lead to quite inefficient (long) codes in practice [29, 38].

Several recently proposed hashing techniques attempt to overcome this problem by finding good *data-aware* hash functions through machine learning. In [29], the authors proposed to use stacked Restricted Boltzmann Machine (RBM) [19, 20], and showed that it was indeed able to generate compact binary codes to accelerate document retrieval. Researchers have also tried the boosting approach to Similarity Sensitive Coding (SSC) [32] and Forgiving Hashing (FgH) [2] — they first train AdaBoost [30] classifiers with similar pairs of items as positive examples (and also non-similar pairs of items as negative examples in SSC), and then take the output of all (decision stump) weak learners on a given document as its binary code. In [36], both stacked-RBM and boosting-SSC were found to work significantly better and faster than LSH when applied to a database containing tens of millions of images. In [38], a new technique called Spectral Hashing (SpH) was proposed. It has demonstrated significant improvements over LSH, stacked-RBM and boosting-SSC in terms of the number of bits required to find good similar items. However, in order to obtain the binary codes for query documents that are unseen before, SpH has to assume that the data are uniformly distributed in a hyper-rectangle, which is apparently very restrictive. In contrast, our proposed Self-Taught Hashing (STH) approach can work with any data distribution so it is much more flexible (see Section 3).

Table 1 gives a brief summary of the above mentioned typical techniques for accelerating similarity search.

3. A REVIEW OF STH

In our paper that is going to appear in the Proceedings of SIGIR-2010 [41], we have proposed a novel Self-Taught Hashing (STH) approach to semantic hashing. As illustrated in Figure 1, STH is a *general* learning framework that consists of two distinct stages: we first find the optimal l -bit binary codes for all documents in the given corpus via unsupervised learning, and then train l classifiers via supervised learning to predict the l -bit code for any query document unseen before. We call the approach “self-taught” because the hash function is learnt from the data that are auto-labelled by itself in the previous stage. We explain these two stages in details as follows.

3.1 Stage 1: Unsupervised Learning of Binary Codes

Given a collection of n documents which are represented as m -dimensional vectors $\{\mathbf{x}_i\}_{i=1}^n \subset \mathbb{R}^m$. Let X denote the $m \times n$ term-document matrix: $[\mathbf{x}_1, \dots, \mathbf{x}_n]$. Suppose that the desired length of code is l bits. We use $\mathbf{y}_i \in \{-1, +1\}^l$ to represent the binary code for document vector \mathbf{x}_i , where the p -th element of \mathbf{y}_i , i.e., $y_i^{(p)}$, is $+1$ if the p -th bit of code is on, or -1 otherwise. Let Y denote the $n \times l$ matrix whose i -th row is the code for the i -th document, i.e., $[\mathbf{y}_1, \dots, \mathbf{y}_n]^T$.

A “good” semantic hashing should be *similarity preserving* to ensure effectiveness. That is to say, semantically similar documents should be mapped to similar codes within a short Hamming distance.

Unlike the existing approaches (such as SpH [38]) that aim to preserve the *global* similarity structure of all document pairs, we focus on the *local* similarity structure, i.e.,

Table 1: Typical techniques for accelerating similarity search.

low-dimensional space	exact similarity search	data-aware	KD-tree, R-tree
		data-oblivious	LSH
high-dimensional space	approximate similarity search	data-aware	LSI, LCH, RBM, SSC, FgH, SpH, STH

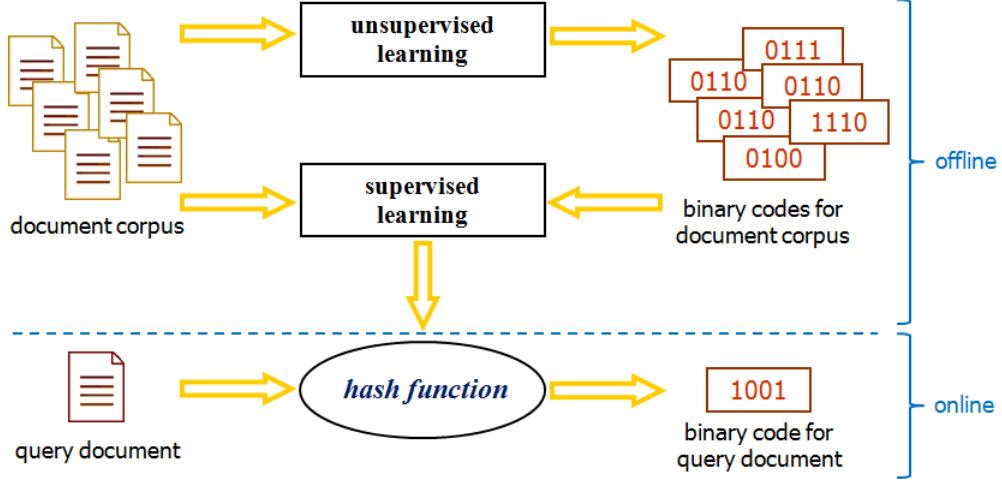


Figure 1: Our recently proposed Self Taught Hashing (STH) technique for accelerating similarity search [41].

k -nearest-neighbourhood, for each document. Since IR applications usually put emphasis on a small number of most similar documents for a given query document [26], preserving the global similarity structure is not only unnecessary but also likely to be sub-optimal for our problem. Therefore, we construct the $n \times n$ local similarity matrix W for the given corpus using the *cosine similarity* [26] or the *heat kernel* [3] as follows:

$$W_{ij} = \begin{cases} \left(\frac{\mathbf{x}_i^T}{\|\mathbf{x}_i\|} \right) \cdot \left(\frac{\mathbf{x}_j}{\|\mathbf{x}_j\|} \right) & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

$$W_{ij} = \begin{cases} \exp\left(-\frac{\|\mathbf{x}_i - \mathbf{x}_j\|^2}{2\sigma^2}\right) & \text{if } \mathbf{x}_i \in N_k(\mathbf{x}_j) \text{ or vice versa} \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

where $N_k(\mathbf{x})$ represents the set of k -nearest-neighbours of document \mathbf{x} . In other words, W is the adjacency matrix of the k -nearest-neighbours graph for the given corpus [3]. A by-product of focusing on such a local similarity structure instead of the global one is that W becomes a sparse matrix. This not only leads to much lower storage overhead, but also brings a significant reduction to the computational complexity of subsequent operations. Furthermore, we introduce a diagonal $n \times n$ matrix D whose entries are given by $D_{ii} = \sum_{j=1}^n W_{ij}$. The matrix D provides a natural measure of document importance: the bigger the value of D_{ii} is, the more ‘‘important’’ is the document \mathbf{x}_i , as its neighbours are strongly connected to it [3].

The Hamming distance between two binary codes \mathbf{y}_i and \mathbf{y}_j (corresponding to documents \mathbf{x}_i and \mathbf{x}_j) is given by the number of bits that are different between them, which can be calculated as $\frac{1}{4}\|\mathbf{y}_i - \mathbf{y}_j\|^2$. To meet the *similarity pre-*

servicing criterion, we seek to minimise the weighted average Hamming distance (as in SpH [38])

$$\frac{1}{4} \sum_{i=1}^n \sum_{j=1}^n W_{ij} \|\mathbf{y}_i - \mathbf{y}_j\|^2 \quad (3)$$

because it incurs a heavy penalty if two similar documents are mapped far apart. After some simple mathematical transformation, the above objective function can be rewritten in matrix form as $\frac{1}{4} \text{Tr}(Y^T L Y)$, where $L = D - W$ is the *graph Laplacian* [7], and $\text{Tr}(\cdot)$ means the matrix trace.

We found the above objective function (3) actually proportional to that of a well-known manifold learning algorithm, Laplacian Eigenmap (LapEig) [3], except that LapEig does not have the constraint $\mathbf{y}_i \in \{-1, +1\}^l$. So, if we relax this discreteness condition but just keep the *similarity preserving* requirement, we can get the optimal l -dimensional real-valued vector $\tilde{\mathbf{y}}_i$ to represent each document \mathbf{x}_i by solving the following LapEig problem:

$$\begin{aligned} \arg \min_{\tilde{\mathbf{Y}}} & \quad \text{Tr}(\tilde{\mathbf{Y}}^T L \tilde{\mathbf{Y}}) \\ \text{subject to} & \quad \tilde{\mathbf{Y}}^T D \tilde{\mathbf{Y}} = I \\ & \quad \tilde{\mathbf{Y}}^T D \mathbf{1} = \mathbf{0} \end{aligned} \quad (4)$$

where $\text{Tr}(\tilde{\mathbf{Y}}^T L \tilde{\mathbf{Y}})$ gives the real relaxation of the weighted average Hamming distance $\text{Tr}(Y^T L Y)$, and the two constraints prevent the collapse into a subspace of dimension less than l . The solution of this optimisation problem is given by $\tilde{\mathbf{Y}} = [\mathbf{v}_1, \dots, \mathbf{v}_l]$ whose columns are the l eigenvectors corresponding to the smallest eigenvalues of the following generalised eigenvalue problem (except the trivial eigenvalue 0):

$$L \mathbf{v} = \lambda D \mathbf{v} \quad (5)$$

We now convert the above l -dimensional real-valued vectors $\tilde{\mathbf{y}}_1, \dots, \tilde{\mathbf{y}}_n$ into binary codes via thresholding: if the p -th element of $\tilde{\mathbf{y}}_i$ is larger than the specified threshold, $y_i^{(p)} = +1$ (i.e., the p -th bit of the i -th code is on); otherwise, $y_i^{(p)} = -1$ (i.e., the p -th bit of the i -th code is off).

A “good” semantic hashing should also be *entropy maximising* to ensure efficiency, as pointed out by [2]. According to the *information theory* [33]: the maximal entropy of a source alphabet is attained by having a uniform probability distribution. If the entropy of codes over the corpus is small, it means that documents are mapped to only a small number of codes (hash bins), thereby rendering the hash table inefficient. To meet this *entropy maximising* criterion, we set the threshold for binarising $\tilde{y}_1^{(p)}, \dots, \tilde{y}_n^{(p)}$ to be the *median* value of \mathbf{v}_p . In this way, the p -th bit will be on for half of the corpus and off for the other half. Furthermore, as the eigenvectors $\mathbf{v}_1, \dots, \mathbf{v}_l$ given by LapEig are orthogonal to each other, different bits $y^{(1)}, \dots, y^{(l)}$ in the generated binary codes will be largely uncorrelated. Therefore this thresholding method gives each distinct binary code roughly equal probability of occurring in the document collection, thus achieves the best utilisation of the hash table.

3.2 Stage 2: Supervised Learning of Hash Function

Mapping all documents in the given corpus to binary codes does not completely solve the problem of semantic hashing, because we also need to know how to obtain the binary codes for query documents, i.e., new documents that are unseen before. This problem, called *out-of-sample extension* in manifold learning, is often addressed using the Nystrom method [4, 11]. However, calculating the Nystrom extension of a new document is as computationally expensive as an exhaustive similarity search over the corpus (that may contain millions of documents), which makes it impractical for semantic hashing. In LPI [17, 16], LapEig [3] is extended to deal with new samples by approximating a linear function to the embedding of LapEig. However, the computational complexity of LPI is very high because its learning algorithm involves eigen-decompositions of two large dense matrices. It is infeasible to apply LPI if the given training corpus is large. In SpH [38], new samples are handled by utilising the latest results on the convergence of graph Laplacian eigenvectors to the Laplace-Beltrami eigenfunctions of manifolds. It can achieve both fast learning and fast prediction, but it relies on a very restrictive assumption that the data are uniformly distributed in a hyper-rectangle.

Overcoming the limitations of the above techniques [4, 11, 17, 16, 38], we propose a novel method to compute the binary codes for query documents by considering it as a supervised learning problem: we think of each bit $y_i^{(p)} \in \{+1, -1\}$ in the binary code for document \mathbf{x}_i as a binary class label (class-“on” or class-“off”) for that document, and train a binary classifier $y^{(p)} = f^{(p)}(\mathbf{x})$ on the given corpus that has already been “labelled” by the above binarised-LapEig method, then we can use the learned binary classifiers $f^{(1)}, \dots, f^{(l)}$ to predict the l -bit binary code $y^{(1)}, \dots, y^{(l)}$ for any query document \mathbf{x} . As mentioned in the previous section, different bits $y^{(1)}, \dots, y^{(l)}$ in the generated binary codes are uncorrelated. Hence there is no redundancy among the binary classifiers $f^{(1)}, \dots, f^{(l)}$, and they can also be trained independently.

In STH, we choose to use the Support Vector Machine (SVM) [22, 31] algorithm to train these binary classifiers. SVM in its simplest form, linear SVM $f(\mathbf{x}) = \text{sgn}(\mathbf{w}^T \mathbf{x})$ consistently provides state-of-the-art performance for text classification tasks [12, 21, 39]. Given the documents $\mathbf{x}_1, \dots, \mathbf{x}_n$ together with their *self-taught* binary labels for the p -th bit $y_1^{(p)}, \dots, y_n^{(p)}$, the corresponding linear SVM can be trained by solving the following quadratic optimisation problem

$$\begin{aligned} \arg \min_{\mathbf{w}, \xi_i} \quad & \frac{1}{2} \mathbf{w}^T \mathbf{w} + \frac{C}{n} \sum_{i=1}^n \xi_i \\ \text{subject to} \quad & y_i^{(p)} \mathbf{w}^T \mathbf{x}_i \geq 1 - \xi_i, \quad i = 1, \dots, n \\ & \xi_i \geq 0, \quad i = 1, \dots, n \end{aligned} \quad (6)$$

Thus, the *predicting* process of STH for a given query document is simply to classify the query document using those l learned classifiers and then assemble the output l binary labels into an l -bit binary code.

4. EXTENSIONS TO STH

4.1 Kernelisation

A prominent advantage of using SVM classifiers in the second stage of STH is that we can easily achieve nonlinear hashing if necessary by rewriting the quadratic optimisation problem (6) into its dual form

$$\begin{aligned} \arg \min_{\alpha} \quad & \sum_{i=1}^n \alpha_i - \frac{1}{2} \sum_{i,j=1}^n y_i^{(p)} y_j^{(p)} \alpha_i \alpha_j \mathbf{x}_i^T \mathbf{x}_j \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \quad i = 1, \dots, n \\ & \sum_{i=1}^n \alpha_i y_i^{(p)} = 0 \end{aligned} \quad (7)$$

and replacing the inner product between \mathbf{x}_i and \mathbf{x}_j by a nonlinear kernel [31] such as the Gaussian kernel:

$$K(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{\|\mathbf{x} - \mathbf{x}'\|^2}{2\sigma^2}\right) \quad (8)$$

Then the p -th bit (i.e., binary feature) of the binary code for a query document \mathbf{x} would be given by

$$f^{(p)}(\mathbf{x}) = \text{sgn}\left(\sum_{i=1}^n \alpha_i y_i^{(p)} K(\mathbf{x}, \mathbf{x}_i)\right) \quad (9)$$

which is a nonlinear mapping.

Here we would like to demonstrate the power of nonlinear hashing through experiments on a couple of synthetic datasets. In the following STH experiments, the parameter $k = 25$ when constructing the k -nearest-neighbours graph for LapEig, and the nonlinear SVM implementation is from LIBSVM [6] with the default parameter values except that the parameter σ for Gaussian kernel (8) is set to a small value 0.1 to facilitate the generation of flexible nonlinear hashing.

Let’s first consider a simple “pie” dataset where the data points are uniformly distributed in a 2D unit circle. There are 1000 data points for training and another 1000 for testing. Suppose that we want to use 16-bit binary codes to encode all the documents, the hash function would need to encompass $l = 16$ mappings each of which corresponds to a bit (i.e., binary feature). Each such mapping, either linear or nonlinear, is essentially a *cut* of the dataset into two

separate partitions: one with the corresponding bit on (1) and the other with the corresponding bit off (0). According to the *entropy maximising* criterion that we have explained in Section 3.1, the ideal cuts should be both balanced and orthogonal. If we choose to use only linear hashing, then it is impossible to keep more than 2 cuts balanced and meanwhile orthogonal. This is because each cut would be just one straight-line dividing the dataset, here the pie, into two partitions, as illustrated in Figure 2. If a straight-line cut is required to be balanced, then it should divide the pie into equal halves, so it must pass the centre point of the pie. Hence, l balanced linear cuts can only divide the pie into $2l = 2 \times 16 = 32$ sectors, which would lead to very inefficient (long) codes. This simple thought experiment reveals the severe limitation of all linear hashing techniques (including LSH, SSC, etc.). In contrast, SpH and STH (employing a nonlinear SVM with Gaussian Kernel) can achieve nonlinear hashing with cuts that are both balanced and orthogonal, as shown in Figure 3(a) and Figure 3(b) where the black colour represents bit 1 and the grey colour represents bit 0. Therefore, using 16-bit binary codes, SpH and STH would be able to effectively divide the pie into $2^l = 2^{16} = 65536$ small pieces. That is the reason why they can generate compact codes for large datasets. Furthermore, the nonlinear cuts of SpH turn out to be just straight-strips, while those generated by STH are much more flexible. The superiority of STH to SpH becomes more apparent when we examine their 16-bit hash functions on the “two-moon” dataset, as shown in Figure 4(a) and Figure 4(b). Even though the dataset is in a very irregular shape, SpH keeps using straight-stripe cuts as the hash function, which results from its restrictive assumption that the data are uniformly distributed in a hyper-rectangle. STH, however, adapts its nonlinear cuts to the real distribution of the data, therefore provides better hash functions.

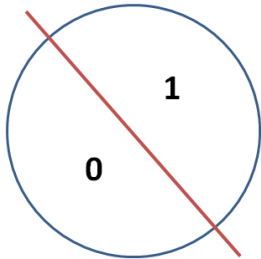


Figure 2: Linear hash functions as straight-line cuts of the dataset.

4.2 Supervision

The k-Nearest-Neighbours (kNN) algorithm [27] is a classic non-parametric machine learning method widely used for data classification. Its efficiency and scalability depend on the speed of similarity search over the large collection of labelled objects, i.e., training examples. By enabling fast similarity search at large scale, semantic hashing techniques like STH makes it feasible to exploit “the unreasonable effectiveness of data” [14] to accomplish traditionally difficult tasks. For example, researchers recently achieved great success in scene completion and scene recognition using millions of images on the Web as training data [15, 36].

The standard STH technique [41] is unsupervised. Al-

though a supervised learning algorithm such as SVM is employed in the second stage of STH, it uses only the *pseudo*-labels input from the previous stage, but still consider the document collection unlabelled.

It is actually easy to extend our STH technique to incorporate the class label information available in the training set of documents for kNN classification. In the first stage of STH — unsupervised learning of binary codes (see Section 3.1) — we make use of the class label information in the construction of k-nearest-neighbour graph for LapEig: a training document \mathbf{x} ’s k-nearest-neighbourhood $N_k(\mathbf{x})$ would only contain k documents in the same class as \mathbf{x} that are most similar to \mathbf{x} . Let **STHs** denote such a supervised version of STH to distinguish it from the standard unsupervised version of STH.

One may ask why it is meaningful to use the kNN algorithm on top of STH that employs SVMs but not use the more powerful SVM algorithm to make classifications directly. The answer is that the kNN algorithm still has its advantages over SVMs in some aspects. As a non-parametric learning algorithm, kNN can yield increasingly complex decision boundaries given more and more training data. More importantly, the learning and prediction time of any multi-class SVM classification mechanism would grow at least linearly in the number of classes, whereas the kNN algorithm has no dependence on the number of classes. For example, if there are 1000 classes, the multi-class SVM approach may need 1000 binary SVM classifiers using the one-vs-rest ensemble scheme, but the STH plus kNN approach using 16-bit binary codes would only require 16 binary SVM classifiers.

We now empirically evaluate supervised version of STH (e.g., STHs), and compare its performance with vanilla STH as well as binarised-LSI [29], LCH [40], and SpH [38] that represents the state of the art (see Section 2). In the following STH and STHs experiments, the parameter $k = 25$ when constructing the k -nearest-neighbours graph for LapEig, and the linear SVM implementation is from LIBLINEAR [13] with the default parameter values.

We have conducted experiments on three publicly available real-world text datasets: Reuters21578¹, 20Newsgroups² and TDT2³.

The Reuters21578 corpus is a collection of documents that appeared on Reuters newswire in 1987. It contains 21578 documents in 135 categories. In our experiments, those documents appearing in more than one category were discarded, and only the largest 10 categories were kept, thus leaving us with 7285 documents in total. We use the ModeApte split here which gives 5228 (72%) documents for training and 2057 (28%) documents for testing.

The 20Newsgroups corpus was collected and originally used for document categorisation by Lang [24]. We use the popular ‘bydate’ version which contains 18846 documents, evenly distributed across 20 categories. The time-based split leads to 11314 (60%) documents for training and 7532 (40%) documents for testing.

The TDT2 (NIST Topic Detection and Tracking) corpus consists of data collected during the first half of 1998 and taken from 6 sources, including 2 newswires (APW, NYT), 2 radio programs (VOA, PRI) and 2 television programs

¹<http://www.daviddlewis.com/resources/testcollections/reuters21578/>

²<http://people.csail.mit.edu/jrennie/20Newsgroups/>

³<http://www.nist.gov/speech/tests/tdt/tdt98/index.htm>

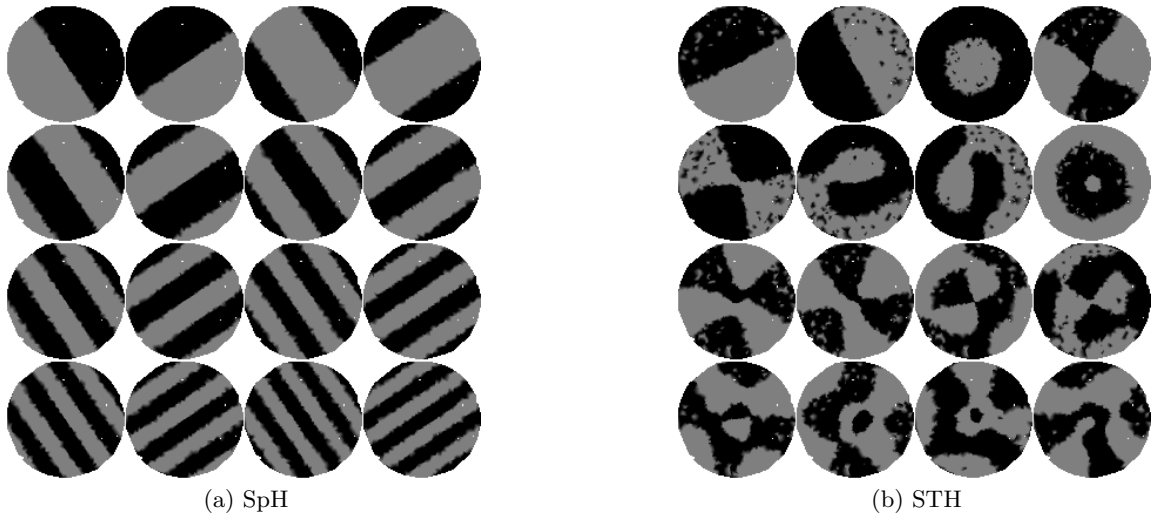


Figure 3: The 16-bit hash function for the pie dataset.

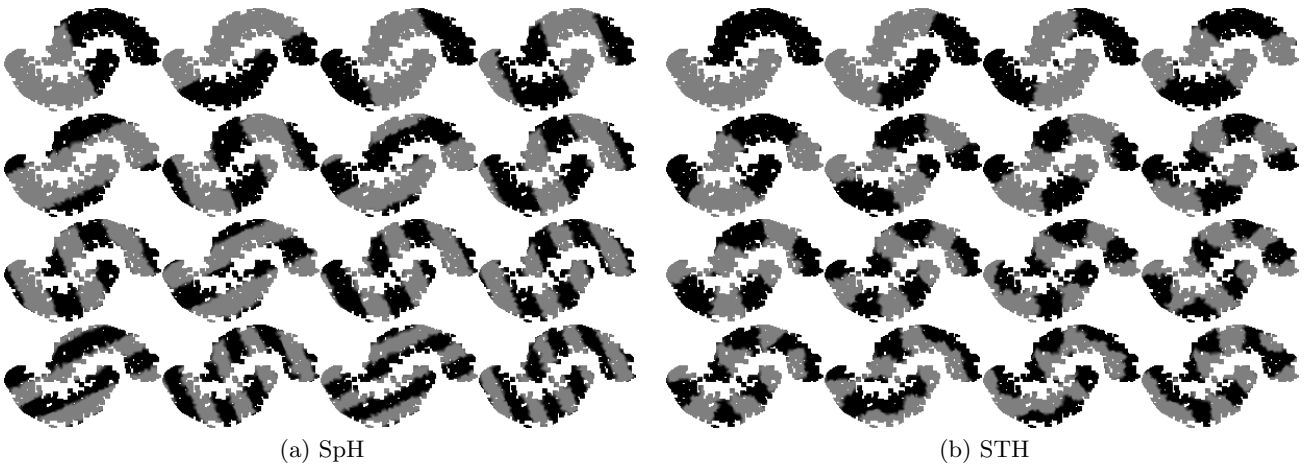


Figure 4: The 16-bit hash function for the two-moon dataset.

(CNN, ABC). It consists of 11201 on-topic documents which are classified into 96 semantic categories. In our experiments, those documents appearing in more than one category were discarded, and only the largest 30 categories were kept, thus leaving us with 9394 documents in total. We randomly selected 5597 (60%) documents for training and 3797 (40%) documents for testing. The averaged performance based on 10 such random selections is reported in this paper.

All the above datasets have been pre-processed by stop-word removal, Porter stemming, and TF-IDF weighting [26].

Given a dataset, we use each document in the test set as a query to retrieve documents in the training set within a fixed Hamming ball of radius 1, and then measure the performance of a semantic hashing technique while varying the code length from 4-bit to 64-bit. Figure 5 shows the averaged *precision-recall curve* [26] for retrieving same-class documents. Figure 6 shows the accuracy for kNN classification using semantic hashing based similarity search, i.e., each test document is classified by the majority label in the Hamming ball centred around it. Here the number of near-

est neighbours for kNN classification is actually not fixed at k , but depends on how many training documents are contained in the specific Hamming ball. It is clear that on all datasets and under both evaluation methodologies, STHs outperforms STH (and other semantic hashing techniques). Using 16-bit codes and Hamming ball radius 1, the performance improvements are all statistically significant (P value < 0.01) according to one-sided micro sign test (s -test) [39].

5. CONCLUSIONS

Our recently proposed STH technique for fast similarity search can be essentially considered as a process of generating a few most informative binary features to represent the documents.

The main contribution of this paper is to present two further extensions to STH, kernelisation and supervision, and moreover show their advantages through experiments on synthetic datasets and real-world datasets respectively. Although both extensions are just minor modifications, they

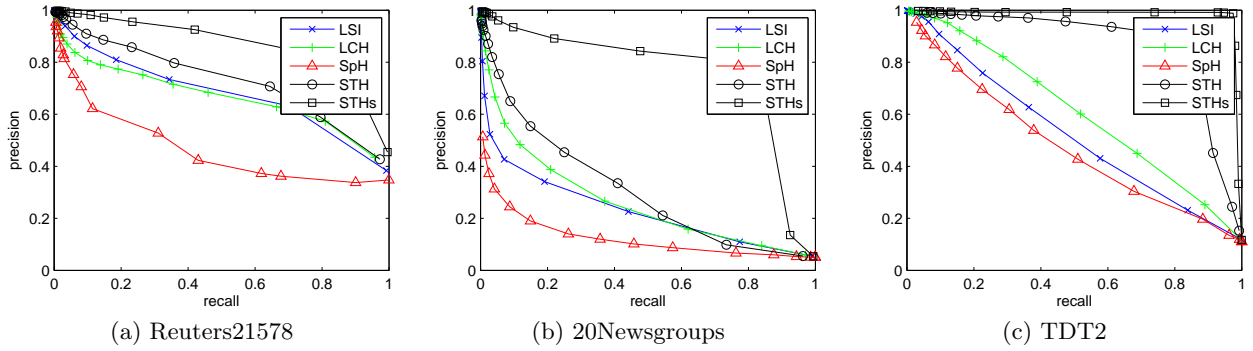


Figure 5: The averaged precision-recall curve for retrieving same-class documents.

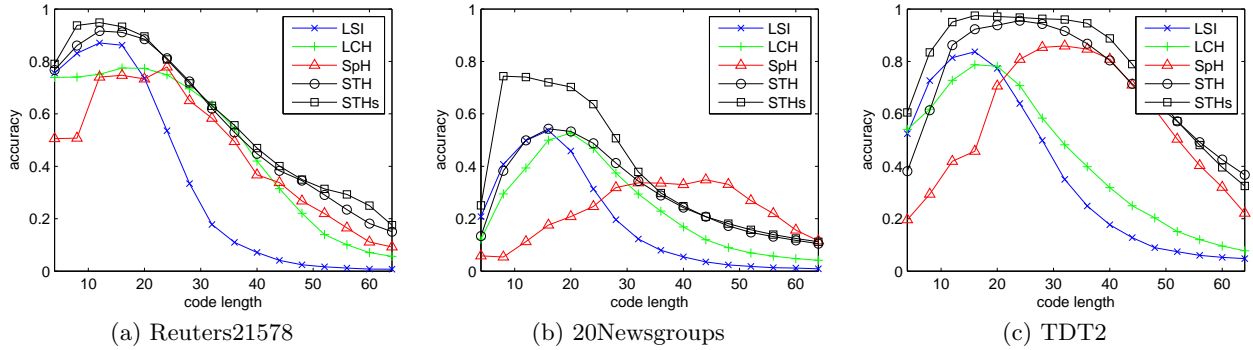


Figure 6: The accuracy for kNN classification using semantic hashing based similarity search.

do work very well in practice to enhance the power or performance of STH.

The future work in our mind includes implementing the STH technique in the MapReduce [9] framework and apply it to large-scale content-based multimedia retrieval [25].

6. ACKNOWLEDGEMENTS

We are grateful to Dr Xi Chen (Alberta) for his valuable discussion and the London Mathematical Society (LMS) for their support of this work (SC7-09/10-6). We would also like to thank the anonymous reviewers for their helpful comments.

7. REFERENCES

- [1] A. Andoni and P. Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In *Proceedings of the 47th Annual IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 459–468, Berkeley, CA, USA, 2006.
- [2] S. Baluja and M. Covell. Learning to hash: Forging hash functions and applications. *Data Mining and Knowledge Discovery (DMKD)*, 17(3):402–430, 2008.
- [3] M. Belkin and P. Niyogi. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural Computation*, 15(6):1373–1396, 2003.
- [4] S. Belongie, C. Fowlkes, F. Chung, and J. Malik. Spectral partitioning with indefinite kernels using the nystrom extension. In *Proceedings of the 7th European Conference on Computer Vision (ECCV)*, pages 531–542, Copenhagen, Denmark, 2002.
- [5] M. W. Berry, S. T. Dumais, and G. W. O’Brien. Using linear algebra for intelligent information retrieval. *SIAM Review*, 37(4):573–595, 1995.
- [6] C.-C. Chang and C.-J. Lin. *LIBSVM: a Library for Support Vector Machines*, 2001.
- [7] F. R. K. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.
- [8] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. MIT Press and McGraw-Hill, 2nd edition, 2001.
- [9] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *Proceedings of the 6th Symposium on Operating System Design and Implementation (OSDI)*, pages 137–150, San Francisco, CA, USA, 2004.
- [10] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society of Information Science (JASIS)*, 41(6):391–407, 1990.
- [11] P. Drineas and M. W. Mahoney. On the nystrom method for approximating a gram matrix for improved kernel-based learning. *Journal of Machine Learning Research (JMLR)*, 6:2153–2175, 2005.
- [12] S. Dumais, J. Platt, D. Heckerman, and M. Sahami. Inductive learning algorithms and representations for text categorization. In *Proceedings of the 7th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 148–155, Bethesda, MD, 1998.
- [13] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874, 2008.
- [14] A. Y. Halevy, P. Norvig, and F. Pereira. The unreasonable effectiveness of data. *IEEE Intelligent Systems*, 24(2):8–12, 2009.
- [15] J. Hays and A. A. Efros. Scene completion using millions of photographs. *ACM Transactions on Graphics (TOG)*, 26(3):4, 2007.
- [16] X. He, D. Cai, H. Liu, and W.-Y. Ma. Locality preserving indexing for document representation. In *Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 96–103, Sheffield, UK, 2004.

- [17] X. He and P. Niyogi. Locality preserving projections. In *Advances in Neural Information Processing Systems (NIPS)*, volume 16, pages 153–160, Vancouver and Whistler, Canada, 2003.
- [18] M. R. Henzinger. Finding near-duplicate web pages: A large-scale evaluation of algorithms. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 284–291, Seattle, WA, USA, 2006.
- [19] G. Hinton, S. Osindero, and Y. W. Teh. A fast learning algorithm for deep belief nets. *Neural Computation*, 18(7):1527–1554, 2006.
- [20] G. Hinton and R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, July 2006.
- [21] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the 10th European Conference on Machine Learning (ECML)*, pages 137–142, Chemnitz, Germany, 1998.
- [22] T. Joachims. *Learning to Classify Text using Support Vector Machines*. Kluwer, 2002.
- [23] Y. Koren. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 426–434, Las Vegas, NV, USA, 2008.
- [24] K. Lang. Newsweeder: Learning to filter netnews. In *Proceedings of the 12th International Conference on Machine Learning (ICML)*, pages 331–339, Tahoe City, CA, 1995.
- [25] M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMCCAP)*, 2(1):1–19, 2006.
- [26] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, 2008.
- [27] T. Mitchell. *Machine Learning*. McGraw Hill, international edition, 1997.
- [28] S. Pandey, A. Broder, F. Chierichetti, V. Josifovski, R. Kumar, and S. Vassilvitskii. Nearest-neighbor caching for content-match applications. In *Proceedings of the 18th International Conference on World Wide Web (WWW)*, pages 441–450, Madrid, Spain, 2009.
- [29] R. Salakhutdinov and G. Hinton. Semantic hashing. *International Journal of Approximate Reasoning (IJAR)*, 50(7):969–978, 2009.
- [30] R. E. Schapire. The boosting approach to machine learning: An overview. In *Nonlinear Estimation and Classification*. Springer, 2003.
- [31] B. Scholkopf and A. J. Smola. *Learning with Kernels*. MIT Press, Cambridge, MA, 2002.
- [32] G. Shakhnarovich, P. A. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *Proceedings of the 9th IEEE International Conference on Computer Vision (ICCV)*, pages 750–759, Nice, France, 2003.
- [33] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 27:379–423, 623–656, 1948.
- [34] B. Stein. Principles of hash-based text retrieval. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 527–534, Amsterdam, The Netherlands, 2007.
- [35] B. Stein, S. M. zu Eissen, and M. Potthast. Strategies for retrieving plagiarized documents. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 825–826, Amsterdam, The Netherlands, 2007.
- [36] A. B. Torralba, R. Fergus, and Y. Weiss. Small codes and large image databases for recognition. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, Anchorage, AK, USA, 2008.
- [37] R. Weber, H.-J. Schek, and S. Blott. A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces. In *Proceedings of 24th International Conference on Very Large Data Bases (VLDB)*, pages 194–205, New York City, USA, 1998.
- [38] Y. Weiss, A. B. Torralba, and R. Fergus. Spectral hashing. In *Advances in Neural Information Processing Systems (NIPS)*, volume 21, pages 1753–1760, Vancouver, Canada, 2008.
- [39] Y. Yang and X. Liu. A re-examination of text categorization methods. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 42–49, Berkeley, CA, 1999.
- [40] D. Zhang, J. Wang, D. Cai, and J. Lu. Laplacian co-hashing of terms and documents. In *Proceedings of the 32nd European Conference on IR Research (ECIR)*, pages 577–580, Milton Keynes, UK, 2010.
- [41] D. Zhang, J. Wang, D. Cai, and J. Lu. Self-taught hashing for fast similarity search. In *Proceedings of the 33rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, Geneva, Switzerland, 2010.