Efficient Implicit Unsupervised Text Hashing using Adversarial Autoencoder

Khoa D. Doan
Dept. of Computer Science
Virginia Tech, Arlington, VA
khoadoan@vt.edu

Chandan K. Reddy
Dept. of Computer Science
Virginia Tech, Arlington, VA
reddy@cs.vt.edu

ABSTRACT

Searching for documents with semantically similar content is a fundamental problem in the information retrieval domain with various challenges, primarily, in terms of efficiency and effectiveness. Despite the promise of modeling structured dependencies in documents, several existing text hashing methods lack an efficient mechanism to incorporate such vital information. Additionally, the desired characteristics of an ideal hash function, such as robustness to noise, low quantization error and bit balance/uncorrelation, are not effectively learned with existing methods. This is because of the requirement to either tune additional hyper-parameters or optimize these heuristically and explicitly constructed cost functions. In this paper, we propose a Denoising Adversarial Binary Autoencoder (DABA) model which presents a novel representation learning framework that captures structured representation of text documents in the learned hash function. Also, adversarial training provides an alternative direction to implicitly learn a hash function that captures all the desired characteristics of an ideal hash function. Essentially, DABA adopts a novel single-optimization adversarial training procedure that minimizes the Wasserstein distance in its primal domain to regularize the encoder's output of either a recurrent neural network or a convolutional autoencoder. We empirically demonstrate the effectiveness of our proposed method in capturing the intrinsic semantic manifold of the related documents. The proposed method outperforms the current state-of-the-art shallow and deep unsupervised hashing methods for the document retrieval task on several prominent document collections.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; *Document representation*; *Content analysis and feature selection*; • **Theory of computation** → **Adversarial learning**; • **Computing methodologies** → **Neural networks**; Unsupervised learning.

KEYWORDS

Hashing, autoencoder, adversarial training, deep learning.

ACM Reference Format:
Khoa D. Doan and Chandan K. Reddy. 2020. Efficient Implicit Unsupervised Text Hashing using Adversarial Autoencoder. In *Proceedings of The Web Conference 2020 (WWW '20), April 20–24, 2020, Taipei, Taiwan.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3366423.3380150

WWW '20, April 20–24, 2020, Taipei, Taiwan

ACM ISBN 978-1-4503-7023-3/20/04.
https://doi.org/10.1145/3366423.3380150

1 INTRODUCTION

One of the crucial challenges associated with mining massive text corpora is to *efficiently* and *effectively* search for documents with **similar** *semantic* content, a problem known as "semantic similarity search" in the information retrieval domain. *Exact similarity search*, which aims to exhaustively find all the relevant documents, is often impractical due to its computational complexity. Specifically, it requires an inefficient linear scan of all documents in the database. The number of documents in the database range in the order of millions (or sometimes even billions). Thus, *approximate similarity search* algorithms that *efficiently* examine a smaller subset of candidate documents provide a principled approach for solving this problem. For instance, in *hashing*, a classic approximate-similarity search approach, we project the original input in the high-dimensional space onto a much smaller locality-preserving *binary* space. Therefore, the full-linear scan is no longer imperative. Compact binary codes are storage-efficient and the search-cost in the original high-dimensional space is reduced to calculating Hamming distances. The binary representation of a document often requires 4 to 8 bytes of storage and computing the pairwise Hamming distance using "bitwise" XOR needs a single CPU instruction.

While hashing methods ensure efficiency, an important challenge is to learn a hash function that *effectively* captures the semantic similarity between a pair of text documents, especially, in the presence of noisy data. Historically, supervised hashing techniques display a better retrieval performance, but they require human-annotated documents that are expensive to obtain on massive-scale datasets [5, 9, 22, 36]. Unsupervised hashing methods do not require supervised signals for training, thus are more suitable for massive-scale corpora [5, 10, 14, 26, 29, 37, 38].

For analyzing text data, majority of the existing unsupervised hashing methods employ shallow, linear hash functions. On the other hand, deep models are capable of learning non-linear hash functions [5, 26, 29, 38]. However, despite the improved performance, they heavily rely on manually-engineered feature vectors such as Term Frequency-Inverse Document Frequency (TF-IDF) [27]. To the best of our knowledge, no prior work in the literature attempts to learn hash functions that capture the semantic and syntactic representation of raw text documents. Although, respectable performance is observed in the context of other text mining problems [20, 25, 39, 40].

In addition to capturing optimal representation of the documents in the hash functions, the learned hash codes should also achieve bit balance/uncorrelation and low-quantization error [35]. Bit-balance

and bit-uncorrelation ensure that a uniform number of training data points are assigned to each hash code. Thus, minimizing the time complexity of the retrieval task in the worst and average cases [13]. Along with low-quantization error, they alleviate the chance of assigning "very" similar data points to different codes. This preserves the original locality structure of the data in the binary space and improves the retrieval precision [10, 13, 16]. In spite of possessing better retrieval performance, existing deep text-hashing methods either ignore these objectives or construct them heuristically. We argue that the retrieval performance is significantly improved by incorporating them in learning the deep hash functions. Furthermore, we prove that it is possible to *implicitly learn the hashing objectives* in an *adversarial-training framework*. Thus, we can eliminate the need to tune several "hyperparameters" when explicitly defining the heuristic hashing objectives in a learning-to-hash model.

To address these aforementioned challenges, we propose *a novel unsupervised **D**enoising **A**dversarial **B**inary **A**utoencoder* (**DABA**) model for the text hashing problem. The proposed DABA model learns a non-linear hash function that captures an optimal representation directly from sequential input data and is robust to the noise in the input data through the application of an adversarially-trained denoising autoencoder. The main contributions of the paper are as follows:

- Propose a novel unsupervised deep generative autoencoder-based hashing model that jointly learns an efficient representation of text documents directly from the raw text and a robust, deep hash function in an end-to-end framework. To the best of our knowledge, this is the first work in text hashing that learns a hash function directly from the raw text data.
- Employ adversarial training procedure; the proposed method introduces a new and efficient alternative to train binary neurons that implicitly generate balanced and discriminative hash codes without introducing additional hyperparameters or complex hashing objective functions. Thus, fulfilling the desired characteristics of an ideal hashing algorithm (as shown in Table 1).
- Propose a novel algorithm to train the adversarial, binary autoencoder that directly estimates the Wasserstein distance from the primal domain by solving the Optimal Transport (OT) problem. The proposed algorithm does not employ the familiar min-max game, which is harder to train, because of the fluctuating generator's cost. Thus, increasing its suitability for large-scale and real-world text hashing problems.
- Demonstrate the effectiveness of our method in large-scale text datasets and demonstrate the superiority of the proposed model over state-of-the-art hashing techniques with both quantitative and qualitative analysis of the performance.

The rest of the paper is organized as follows. We discuss the related work in Section 2. In Section 3, we describe the details of our proposed method. Finally, we present quantitative and qualitative experimental results in Section 4 and conclude in Section 5.

2 RELATED WORK

In this section, we describe two lines of research that are closely related to the proposed work: hashing and deep learning.

2.1 Similarity Search and Hashing

2.1.1 Hashing. The most representative class for semantic similarity search is hashing. Hashing techniques can be broadly classified into supervised [5, 9, 22, 36] and unsupervised hashing [5, 10, 14, 26, 37, 38]. Although supervised hashing methods demonstrate better performance [5, 29], they require human-annotated documents that are expensive to obtain on massive-scale datasets, which are increasingly prevelant in the current scenario. Scarcity of labeled training data introduces the problem of train/test distribution mismatch and poor local optima converge in supervised learning frameworks. Therefore, their search performance degrades significantly.

Unsupervised hashing approaches address these problems by learning hash functions without any supervised signal. Thus, these methods are more suitable for large-scale corpora. Predominantly, unsupervised hashing methods include shallow and deep models. Locality sensitive hashing (LSH) approaches, which learn the hash functions using random projection, are popular shallow models with appealing theoretical properties [8, 21, 30, 34].

Data-dependent shallow-hashing methods, notably Spectral Hashing (SpecHash) [37] and Semantic Hashing (SemHash) [26], demonstrate significant performance improvements over LSH. SpecHash learns compact, balanced and uncorrelated-bit codes by solving the Eigenvector problem. However, the method strongly assumes that the data points are uniformly distributed in a hyper-rectangle, which restricts its application potential. Spherical Hashing (SpheHash) [14], on the other hand, employs hypersphere-based partitioning to achieve better and coherent quantization. Iterative Quantization (ITQ) additionally minimizes the quantization error to generate hash functions that better preserves the original locality structure of the input data. However, these shallow models are linear and hence cannot be applied to real-world high-dimensional datasets.

Unsupervised, deep-hash models [5, 26, 29, 38], display remarkable performance improvements compared to the shallow models by learning non-linear hash functions. Self-taught hashing (STH) [38] decomposes the algorithm into learning the hash function (similar to that of SemHash) and learning a hash function on unseen data using a supervised learning method. Variational Deep Semantic Hashing (VDSH) [5] employs variational autoencoders to learn a generative model to reconstruct the original documents. Neural Architecture for Semantic Hashing (NASH) [29] adopts a similar variational autoencoder architecture as that of VDSH but models the hashing codes as Bernoulli latent-variable. These existing deep hash models, in spite of having better performance, do not exploit the latent semantic and syntactic dependencies in the sequential text data to learn the hash functions. For example, both VDSH and NASH employ the hand-crafted TF-IDF input. To the best of our knowledge, our proposed model is the first end-to-end hashing model that captures the structured representation of raw text documents.

2.1.2 Characteristics of Hash Functions. Prevalent hashing approaches minimize the training objective as follows:

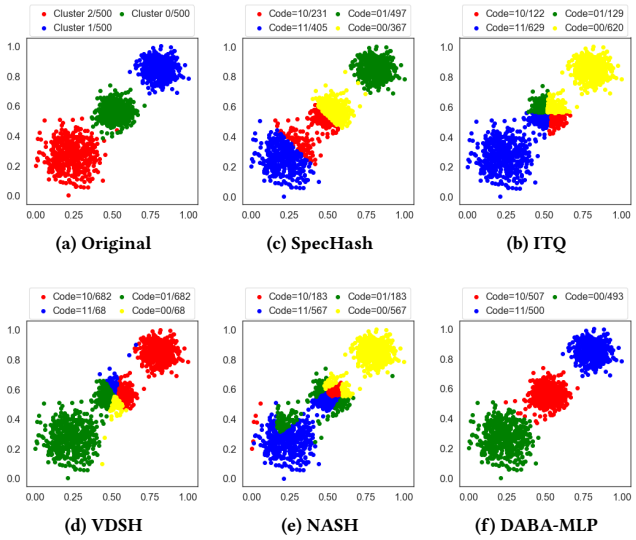
$$\min_{f} E_{x\sim D_x} L(x, f(x)) + E_{x\sim D_x} \sum_{k} \lambda_i \times H_k(f(x)) \qquad (1)$$

Table 1: Characteristics of hashing algorithms along with the representative methods. Our proposed method satisfies all these characteristics in an end-to-end framework without introducing complex cost functions and hyperparameters. ○ indicates the method achieves the objective, but the procedure is ad-hoc.

	LSH [21]	SpecHash [37]	SpheHash [14]	ITQ [10]	SemHash [26]	STH [38]	VDSH [5]	NASH [29]	DABA (Ours)
Non-linear	×	×	×	×	✓	✓	✓	✓	✓
Low Quantization Error	×	×	×	✓	○	✓	×	×	✓
Bit Balance	×	✓	✓	✓	○	✓	✓	✓	✓
Bit Uncorrelation	×	✓	✓	✓	✓	✓	✓	✓	✓
Data Dependent	×	✓	✓	✓	✓	✓	✓	✓	✓
Learn from Raw Text	×	×	×	×	×	×	×	×	✓
Implicit Generative Model	×	×	×	×	×	×	×	×	✓
Robustness to Noisy Data	×	×	×	×	×	×	×	×	✓

where D_x is the data distribution, $L(x, f(x))$ is the locality-preserving loss of the hash function $f(x)$ and $H_k(f(x))$ is a hashing objective with λ_k as its corresponding hyperparameter. Examples of H_k are:

- *Bit balance* [37]: each bit has the same chance of being 0 or 1.
- *Bit uncorrelation* [37]: different bits are uncorrelated.
- *Low quantization error* [10]: low information loss by encoding the real-valued representation space by the binary, discrete space.



(a) Original (c) SpecHash (b) ITQ

(d) VDSH (e) NASH (f) DABA-MLP

Figure 1: A synthetic example of 1500 data points generated from three Gaussian clusters – 500 data points per cluster (Figure 1a). Each method learns a 2-bit hash function without using the supervised cluster-labels. An optimal hash function achieves "code balance" and recovers the original cluster assignment as much as possible.

In hashing, including H_k's are important for improving the retrieval performance. For example, bit-balance and bit-uncorrelation encourage "code-balance", i.e., the condition where a uniform number of training data points is assigned to each hash code. Code-balance minimizes the time complexity of the worst and average cases, thus, improving the retrieval efficiency [13]. Furthermore, code-balance and low-quantization error alleviate the chance of assigning "very" similar data points to codes that have large Hamming distances. Hence, they better preserve the original locality structure of the data [10, 13, 16]. Preserving locality improves the retrieval precision. It is critical that these three objectives are jointly optimized in any efficient hashing algorithm. For example, without the bit-uncorrelation constraint, an algorithm can still achieve bit balance. A trivial example is assigning half the number of data points to a code with all 0-bits and the remaining half to a code with all 1-bits. However, the quality of the hash codes are apparently poor because there are only two possible codes regardless of the number of bits.

Figure 1 depicts a concrete example of data points that are independently drawn from three two-dimensional Gaussian clusters (500 data points per cluster). Figure 1a illustrates the original generated data points. SpecHash and ITQ, the shallow-hashing methods, could not learn a non-linear separation of the data points (Figures 1b-1c). On the other hand, VDSH and NASH, the deep-hashing methods, are able to learn non-linear hash functions but they do not achieve code-balance (the number of data points per hash code has high variance, as shown in Figures 1d and 1e). The reason is that they ignore bit-balance, bit-uncorrelation and low-quantization error. Only the proposed model, DABA-MLP, is able to recover the original cluster's data point assignments almost perfectly and achieve code-balance. DABA-MLP is a non-linear technique and, more importantly, is constrained to generate both bit-balance and bit-uncorrelation. Furthermore, distinct from existing approaches, the hashing objectives (or H_k) are **implicitly optimized** (using adversarial training) without tuning additional hyperparameters.

A summary of the characteristics of different text hash algorithms is given in Table 1.

2.2 Semantic Hashing and Deep Learning

SemHash [26] is one of the early deep hash models. SemHash learns a binary vector representation of the input data using Restricted Boltzmann Machine (RBM) tuned autoencoders. The hashed binary vectors capture the semantic dependency of the input data in a lower-dimensional space. However, RBM is harder to train for achieving a generative power similar to that of a denoising

Table 2: Notations used in this paper.

Notation	Description
$x, \tilde{x}, \hat{x}$	the original, corrupted and reconstructed input
$x^{(i)}$	the i^{th}-indexed sample input document
$\hat{w}_{x,l}, w_{x,l}$	predicted and actual words at position l in document x
$\hat{x}_l$	predicted word vector at position l in document x
D_x	data distribution
b, c	sigmoid-representation vector and the corresponding thresholded binary-code vector
z	the sampled input (from $B_p(z)$) to the discriminator
z_i	the i^{th} component of the vector z
$b^{(i)}, z^{(j)}$	the representation of the i^{th}-indexed document and the j^{th}-indexed Bernoulli sample, respectively
$q(b)$	posterior distribution of b
f, g	encoding and decoding functions
$W_e[v]$	output (of dimension e) of the embedding lookup of word v
L_A	reconstruction loss function for the autoencoder
L_M	distribution-matching loss (i.e., divergence)
Θ_E, Θ_D	parameters of the encoder (generator) and the decoder

autoencoder [2]. Furthermore, denoising autoencoders produce noise-invariant low-dimensional representations of the data leading to more robust hash functions [33].

To encourage the code layer to be "binary", SemHash regularizes the sigmoid units with deterministic Gaussian noise during training. This introduces additional hyper-parameters that increase the complexity of the learning process. Although, SemHash and similar semantic models [36, 38] do not require additional hashing objective functions such as the quantization or bit-entropy loss (i.e., is H_k's), the sigmoid layer outputs more activation values near 0 than near 1 (because of the asymmetry between 0 and 1 of the RBM's energy function). Thus, selection of a threshold value becomes heuristic and can be suboptimal. Our DABA training squashes activations of the sigmoid layer almost equally near 0 and 1, achieving "bit balance" [35] (see Figure 4) and encouraging the choice of the same threshold value (0.5) in all datasets.

2.2.1 Representation Learning. Recurrent neural network (RNN) provides a natural and conceptual way to capture the characteristics of sequential data [11]. RNNs have been successfully applied in several domains [25, 41]. In [20], the authors proposed a method to compress a sequence into its vector representation. Similar to RNN, Convolutional Neural Networks (CNN) are also proposed for learning an efficient representation that captures semantic and syntactic structures of text documents for various problems [39, 42]. To the best of our knowledge, there is no prior work in the text domain to simultaneously learn document representation and an efficient hash function by directly exploiting such structures from raw textual data.

2.2.2 Generative Adversarial Network and Adversarial Training. Generative Adversarial Network (GAN) has recently gained popularity due to its ability to generate realistic samples from the data distribution [12]. A prominent feature of GAN is its ability to "implicitly" *match* outputs of a deep network to a pre-defined distribution using the adversarial training procedure. In fact, adversarial training has been selected in several works as a regularization of the latent representation of the autoencoders [17, 24]. [17] report that adversarially-trained autoencoders efficiently learn the intrinsic manifold of the data, especially, structured data without overfitting and poor local minima convergence. Furthermore, in contrast to other generative models such as Variational Autoencoders [19], it is possible to learn an "optimal" hash function implicitly by adversarial regularization of the hash function with a "binary" distribution to guide the learning process without imposing any additional constraints H_k's [7]. However, training adversarial autoencoders remains challenging and inefficient because of the alternating-optimization procedure (min-max game) between the generator and the discriminator. For example, [7] employs the original min-max GAN objective [12], which suffers from mode-collapse and vanishing gradient [1]. Moreover, in min-max optimization, the generator's loss fluctuates during training instead of "descending", making it extremely challenging to know when to stop the training process. Wasserstein GAN overcomes a few of these limitations (specifically, mode-collapse and vanishing gradient) [1]. However, it approximates the Wasserstein distance by employing the Kantorovich-Rubinstein dual, thus, resulting in a similar min-max game between the generator and the critic. [31] proposes Wasserstein Autoencoder but still employs the min-max game. We propose to directly approximate the Wasserstein distance from the primal direction by solving the Optimal Transport problem [4].

3 PROPOSED MODEL

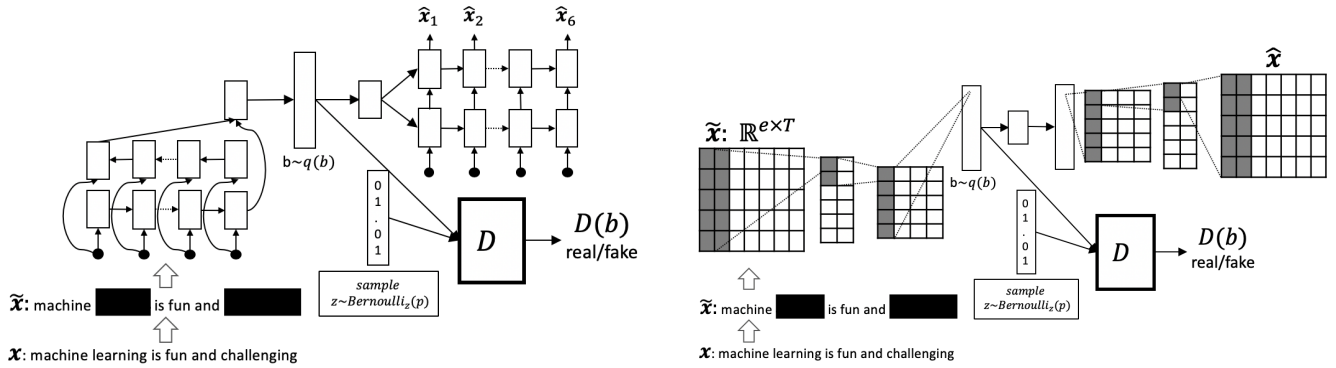
3.1 Problem Formulation

Given a query document e, the *similarity search problem* is to find a set of K similar documents $S(e)$ from a database $X = \{x^{(1)}, x^{(2)}, ..., x^{(N)}\}$ of N documents such that, given $sim(x, y)$ as a pre-defined similarity function that measures the similarity between two documents x and y, we have $sim(e, x) \leq sim(e, y)$ for all $x \in S(e), y \in X \setminus S(e)$.

In hashing, the goal is to find a hash function $c = f(x)$ whose output c, called "hash code", is a low-dimensional, binary vector in $\{0, 1\}^{|c|}$, where the notation $|c|$ is the cardinality (or dimension) of c, such that the relationship between x and y, through $sim(x, y)$, is preserved in $\{0, 1\}^{|c|}$. Learning a discrete-output function f is intractable [35], thus we instead learn a continuous function $b = f(x) \in \mathbb{R}^{|b|}$, from which c is obtained by thresholding b. In the text domain, each data point x or y is a vector in a high-dimensional space $\mathbb{R}^n$, such as the TF-IDF vector, or a sequence of one-hot vectors of the words representing the raw text document. $|b|$ is typically small and takes a value between 16 to 128. The notations used in this paper are given in Table 2. In the following sections, we will discuss the components of the proposed method.

3.2 Binary Autoencoder

Given a dataset $X = \{x|x \sim D_x\}$, a binary autoencoder defines an encoding function $f : x \rightarrow b$ that maps each data point x



(a) DABA-RNN (b) DABA-CNN

Figure 2: Overall system architecture of the proposed Denoising Adversarial Binary Autoencoder (DABA) model.

into a point $b \in \mathbb{R}^{|b|}$ in the coding space and a decoding function $g : b \rightarrow \hat{x}$ that recovers x from b. When the encoder's output layer (i.e. the last layer) is sigmoid, thresholding the activations b (e.g. at 0.5) will return a binary code vector $c \in \{0,1\}^{|b|}$ which is considered as the discrete hash code of the input x. Modeling the hash function as a structured, deep encoder, captures complex latent document representation from the raw text. Therefore, we define the binary autoencoder with the following network structure:

- *An encoder function* $f : x \rightarrow b$ maps a sequence of text x into a low-dimensional vector b. In our work, we model f as either a bidirectional RNN or a convolutional neural network (CNN), followed by a single-layer projection onto the sigmoid layer:
  - The RNN encoder (Figure 2a) is a multi-layer, bidirectional RNN with Long-Short-Term-Memory (LSTM) Units. We observe similar performance using more than one layer and other types of hidden units, such as Gated Recurrent Units [6].
  - The CNN encoder (Figure 2b) consists of multiple convolutional layers. For example, in our experiments on 20NewsGroups and Reuters datasets, given an embedding input $\tilde{x} \in \mathbb{R}^{e\times T}$ where T is the max length of input sequences and e the word's embedding dimension, we find that the best performance is achieved by using two convolutional layers with Rectified Linear Unit (ReLU) activation function with kernels $\{5\times e, 5\times 1\}$, strides $\{2, 2\}$ and numbers of filters $\{300, 600\}$, resulting in $\{c_1\times 300, c_2\times 600\}$ feature maps, respectively (where c_1 and c_2 are the dimensions of the feature maps at layer 1 and layer 2, respectively). This is followed by a max-out layer of 300 filters of $c_2 \times 1$ kernel and a stride of 1.
- *A decoder function* $g : b \rightarrow \hat{x}$ that reconstructs the input x as $\hat{x}$. The function g is modeled as a single-layer linear projection followed by an RNN (for an RNN-encoder) or a CNN (for a CNN-encoder) to reconstruct the input x:
  - The RNN decoder (Figure 2a) is a multi-layer LSTM network whose output $\hat{x_l}$ at position l in the reconstructed sequence $\hat{x}$ of L words is the predicted softmax probability of word $\hat{w}_{x,l}$ at position l in the reconstructed sequence $\hat{x}$, defined as

$$p(\hat{w}_{x,l} = v) = \phi(\hat{h}_l, W_e[v]) \tag{2}$$

    where $\hat{h}_l$ is the hidden state representation of the LSTM decoder at time-step l, $W_e[v]$ is the embedding of word v, and $\phi(v_1, v_2)$ is the softmax function.
  - The CNN decoder (Figure 2b) consists of deconvolutional (convolutional transpose) layers to decode the hidden representation back to the original embedded input [42]. The probability that the predicted word at position l in the reconstructed sequence $\hat{x}$, $\hat{w}_{x,l}$, is v can be defined as follows:

$$p(\hat{w}_{x,l} = v) = \frac{\exp(\tau^{-1} cos(\hat{x}_l, W_e[v]))}{\sum_{v' \in V} \exp(\tau^{-1} cos(\hat{x}_l, W_e[v']))} \tag{3}$$

    where τ is the Gumbel-softmax parameter, V is the vocabulary, and $cos(.)$ is the cosine distance function. We use $\tau = 0.01$ in our experiments.

It is important to note that modeling the encoder/decoder with other network architectures such as Attention [32] is a straightforward extension to our paper. The objective function of the binary autoencoder described above can be written as the word-wise negative log-likelihood, averaged across all documents, as follows:

$$L_{\mathrm{A}} = E_{x\sim D_x}\left[\frac{1}{L}\sum_{l=1..L} -\log\, p(\hat{w}_{x,l} = w_{x,l})\right] \tag{4}$$

where $w_{x,l}$ is the actual word at position l in the input document x.

3.3 Denoising Binary Autoencoder

As discussed in Section 2, denoising autoencoders achieve generative power similar to that of RBM-based autoencoders which are arduous to train [2]. Moreover, the denoising component allows the autoencoder to robustly learn the hash function of corrupted data by utilizing the semantic relationship of uncorrupted words in the text. Therefore, we propose to extend the binary autoencoder, as defined in Section 3.2, with a denoising component.

Specifically, for each word-position in a document, we randomly remove the word with probability p_d. p_d is, therefore, the fraction of the words of the input sequences that need to be removed. The input to the model is the corrupted input $\tilde{x}$ and the model attempts to recover x. Denoising autoencoders are also known to help in

learning a smooth, low-dimensional manifold of the data similar to that of RBMs [3]. We call this model Denoising BA (DBA).

3.4 Denoising Adversarial Binary Autoencoder

An autoencoder defines an encoding distribution $q(b|x)$ and a decoding distribution $p(x|b)$. Consequently, the encoding step imposes a posterior distribution on the coding space as follows:

$$q(b) = \int_x q(b|x)D_x(x)dx \tag{5}$$

The posterior distribution is also synonymous to the hash's output distribution. Therefore, if we force $q(b)$ to "agree" with a binary-like distribution, such as a discrete "Bernoulli prior", that has all desired characteristics of a good hashing function as discussed in Section 2, the adversarial training procedure will force the encoder function $f(x)$ to generate binary codes with the same desired characteristics. Specifically, an input vector z is a sample from a Bernoulli distribution $B_p(z)$ with a parameter p if each of its components z_i is independently drawn from the Univariate Bernoulli distribution with parameter p. Given $z \sim B_p(z)$, we regularize the posterior distribution $q(b)$ to be similar to $B_p(z)$. The adversarial binary autoencoder (ABA) is trained with dual objectives as follows:

$$\min_{\Theta_E, \Theta_D} L_A + L_M \tag{6}$$

where Θ_E is encoder's parameters, Θ_D is the decoder's parameters and

(1) L_A is the reconstruction error from the encoding space b in the decoding step, defined in Section 3.2.
(2) L_M is the regularization term. It is the divergence of the encoding posterior distribution $q(b)$ and the Bernoulli-prior distribution $B_p(z)$ where p is set to 0.5 (hereafter, we denote $B_{0.5}(z)$ as $B(z)$). An intuitive explanation of the regularization is that every component of b will learn to divide as optimal as possible the original space into two halves (thus its activation has about 50% chance of being closer to 1 or closer to 0 – a "bit balance" [35]), where the points in each half are more similar to those in the same half compared to those in the other half – "bit uncorrelation" [35].

We solve Equation (6) by playing a game between the generator and discriminator, equivalently minimizing the Jensen Shannon divergence [17]; in other words, the training objective is as follows:

$$\min_{\Theta_E, \Theta_D} \left(L_A + \max_{\Theta_{Disc}} L_M \right) \tag{7}$$

where Θ_{Disc} is the parameters of the discriminator. The authors of [31] generalize this framework to Wasserstein distance, but it employs a similar min-max game. In this paper, we follow the primal direction of estimating the Wasserstein distance, thus, removing the min-max game. Specifically, in the primal domain, Wasserstein distance is defined as follows:

$$W(q(b), B(z)) = \inf_{\gamma \in \Pi(q(b), B(z))} \int_{(b,z)\sim\gamma} p(b,z)d(b,z)dbdz \tag{8}$$

where $\Pi(q(b), B(z))$ is the set of all possible joint distributions of b and z whose marginals are $q(b)$ and $B(z)$, respectively, and $d(b,z)$ is the cost of transporting one unit of mass from b to z. This approach is equivalent to solving the optimal transport (OT) problem, whose objective is to find the optimal transport plan to move masses from the generated distribution $q(b)$ to the true distribution $B(z)$. Given two finite samples of N examples of b and N examples of z, the empirical transport cost can be formulated as the following Linear Programming (LP) problem:

$$\hat{W}(q(b), B(z)) = \min_M \sum_i^N \sum_j^N M_{i,j} d(b^{(i)}, z^{(j)}) = \min_{\Theta_E} M \odot D \tag{9}$$

where M is the assignment matrix, D is the cost matrix where $D_{ij} = d(b^{(i)}, z^{(j)})$ and $\odot$ is the Hadamard product. The LP formulation has the following constraints:

$$\sum_i^N M_{i,j} = 1, \forall j = 1, ..., N \tag{10}$$

$$\sum_j^N M_{i,j} = 1, \forall i = 1, ..., N \tag{11}$$

$$M_{i,j} \in \{0, 1\}, \forall i = 1, ..., N, \forall j = 1, ..., N \tag{12}$$

It is important to note that both $M_{i,j}$ and $d(b^{(i)}, z^{(j)})$ are functions of Θ_E. *The "distribution matching cost" L_M in Equation (6) is equivalent to $\hat{W}(q(b), B(z))$.* Given the optimal assignment matrix $M*$ that solves this LP problem and a learning rate α, we can update the parameters of the generator, Θ_E, as follows:

$$\Theta_E^{\text{new}} = \Theta_E - \alpha \frac{\partial \hat{W}^*(q(b), B(z))}{\partial \Theta_E} \tag{13}$$

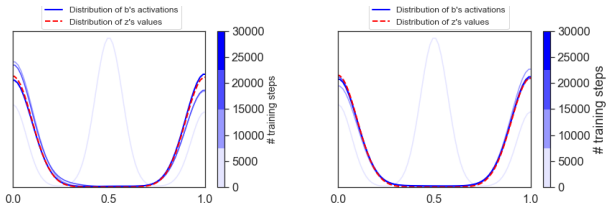
$$\frac{\partial \hat{W}^*(q(b), B(z))}{\partial \Theta_E} = \frac{\partial M^*}{\partial \Theta_E} \odot D + M^* \odot \frac{\partial D}{\partial \Theta_E} \tag{14}$$

While $M*$ depends on Θ_E, it is known that a small perturbation of Θ_E does not change the transport plan; that is $\frac{\partial M^*}{\partial \Theta_E} = 0$. Thus, we update the generator using only the second term in Equation (14), $M^* \odot \frac{\partial D}{\partial \Theta_E}$, which is easy to compute given a differentiable distance function $d(b,z)$. In our work, we use the Euclidean distance where $d(b,z) = ||b - z||_2$.

Minimizing L_M drives the distribution $q(b)$ closer to that of $B(z)$. Intuitively, this is assigning each data point b to the closest sample z in the Euclidean space (given that $d(b,z)$ is the Euclidean distance). Since the Bernoulli-data generating process samples each component z_k of z independently (hence, each bit is uncorrelated) from a Univariate Bernoulli distribution with probability 0.5 (hence, each bit has 0.5 probability of being 1), matching $q(b)$ against the $B(z)$ is exactly equivalent to maximizing bit-uncorrelation and bit-balance codes after the assignment. The assigned z of the vector b can also be viewed as its quantized vector c. In other words, c is approximately the same as z. Consequently, it minimizes the quantization error when L2-norm cost $d(b,z) = ||b - z||_2$ is applied (because the quantization error is defined as $||b - c||_2$).

The implicit distribution-matching process between b and z is illustrated in Figure 3. At the beginning of the training process, the distribution of b-neurons' output (or activations) is a Gaussian (a result of uniform-random weights' initializations). However, after

several training steps, the distribution of b's activations become very similar to that of the sampled values of z from $B(z)$.



(a) Training Data (b) Validation Data

Figure 3: Distribution of activations of b after several training steps and distribution of values of z, sampled from $Bernoulli_z(0.5)$, of DABA created on the 20Newsgroup dataset.

ALGORITHM 1: DABA Model Training

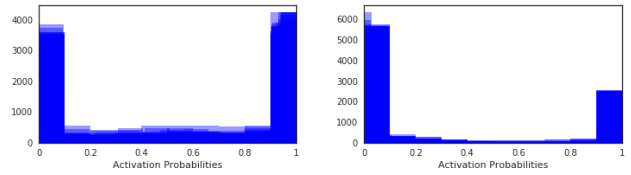
Input: Training data X,
Denoising parameter p_d,
Binary code size $|b|$,
Bernoulli sampling parameter p,
Number of training iterations K.
Output: $\{\Theta_E\}$ parameters of the encoder
for *number of training iterations* K **do**

- Sample a minibatch of m examples $\{x^{(1)}, ..., x^{(m)}\}$.
- Randomly drop elements of $x^{(i)}$ with probability p_d to $\tilde{x}^{(i)}$.
- Sample m vectors $\{z^{(1)}, ..., z^{(m)}\}$ where $z^{(i)} \sim B_p(z)$.
- Update the autoencoder parameters, Θ_E and Θ_D by minimizing L_A in Eq. (4).
- Solve the LP problem given in Eq. (9) to find M^*.
- Update the encoder parameters Θ_E using the update rule described in Eqs. (13) and (14).

end

We call this model as Denoising Adversarial BA (DABA). The overall architecture of our DABA-RNN and DABA-CNN models is shown in Figure 2. The adversarial training algorithm is described in Algorithm 1. For training our model, we adopt the standard mini-batch stochastic gradient descent procedure, using Adam optimizer [18] with $p = 0.5$.

Figure 4 depicts the distribution of the activations (b) of encoder's output layer on the 20NewsGroups dataset for Semantic Hashing and DABA with TF-IDF input [27] and multi-layer perceptrons (MLP) as encoder/decoder (called DABA-MLP with similar network architecture as that of Semantic Hashing). In both methods, we observe that the activations squashed out to the borders near 0 and 1. DABA (shown in Figure 4(a)), however, learns to output activations that are more equally squashed out than those of Semantic Hashing (shown in Figures 4(b)). This allows us to simply pick 0.5 as a threshold for binarizing the activations, as compared to a heuristic suggestion of 0.1 for Semantic Hashing [26].



(a) DABA-Epoch 60 (b) SM-Epoch 60

Figure 4: Activations (values of b) of DABA and Semantic Hashing (SM) after 60 epochs.

4 EXPERIMENTAL RESULTS

4.1 Datasets Used

We utilize the following datasets in the performance evaluation experiments of the proposed DABA model.

- **20NewsGroups** [1]: A collection of 18,821 newsgroup documents categorized uniformly into 20 different newsgroups. The data is split into 90% of the documents that serve as the database and 10% of the documents for querying.
- **Reuters** [2]: A collection of 12,902 Newswire stories provided by Reuters, Ltd. and organized into 135 categories. We randomly split 90% of the documents as the database and the remaining 10% of the documents for querying.
- **DBpedia** [40]: A collection of 630,000 documents classified into 14 non-overlapping ontology classes. The data is split into 560,000 training documents that serve as the database and 70,000 testing documents for querying.

4.2 Evaluation Procedure

For evaluating the performance of the proposed model, we follow the standard mechanism that is widely accepted in the context of the ranking problem- the **precision** metric at various threshold values m ($P@m$):

$$\text{precision@}m = \frac{|\text{retrieved, relevant documents}|}{|\text{retrieved documents}|} \quad (15)$$

where the threshold m is the number of top ranked candidate documents, based on their Hamming distances to the query document, obtained by the algorithms for the retrieval task. For the experiments, we set m to be 10, 50, and 100.

In addition to this, we calculate the limited areas under the Precision-Recall curves (equivalent to limited mean average precision – or MAP) up to a certain threshold m, called MAP@m for different methods [28]. In our experiments, we report MAP@1000.

4.3 Comparison Methods

We compare the performance of the proposed method with various representative similarity-search methods.

- *Locality Sensitive Hashing* (**LSH**) [21]: the popular data-independent, shallow hashing method using random projection.

[1] http://qwone.com/~jason/20Newsgroups/
[2] http://www.daviddlewis.com/resources/testcollections/reuters21578/

Table 3: Performance comparison of different methods using standard metrics (precision at various threshold values m and mean average precision) for the document retrieval task on various text datasets.

	20Newsgroup				Reuters				DBpedia			
	P@10	**P@50**	**P@100**	**MAP**	**P@10**	**P@50**	**P@100**	**MAP**	**P@10**	**P@50**	**P@100**	**MAP**
LSH	0.456	0.388	0.301	0.259	0.455	0.429	0.386	0.389	0.507	0.485	0.417	0.296
SpecHash	0.511	0.448	0.382	0.307	0.782	0.719	0.651	0.429	0.519	0.455	0.431	0.323
SpheHash	0.512	0.455	0.385	0.316	0.752	0.710	0.664	0.421	0.463	0.427	0.408	0.304
ITQ	0.524	0.457	0.384	0.320	0.799	0.731	0.667	0.430	0.516	0.457	0.422	0.336
SemHash	0.520	0.435	0.390	0.322	0.780	0.703	0.650	0.462	0.553	0.503	0.452	0.313
STH	0.523	0.501	0.565	0.328	0.817	0.798	0.755	0.476	0.568	0.506	0.483	0.312
VDSH	0.552	0.550	0.431	0.339	0.839	0.815	0.775	0.495	0.634	0.531	0.475	0.324
NASH	0.579	0.544	0.539	0.349	**0.867**	0.840	0.799	0.501	0.610	0.557	0.505	0.331
DBA	0.491	0.511	0.562	0.301	0.810	0.788	0.679	0.471	0.596	0.526	0.479	0.325
DABA-MLP	0.559	0.542	0.565	0.339	0.836	0.831	0.772	0.497	0.629	0.541	0.511	0.341
DABA-RNN	0.628	0.579	0.565	**0.366**	0.859	0.833	**0.830**	**0.520**	0.641	0.576	**0.539**	0.348
DABA-CNN	**0.639**	**0.612**	**0.590**	0.341	0.860	**0.861**	0.820	0.509	**0.655**	**0.598**	0.537	**0.358**

- *Spectral Hashing* (**SpecHash**) [37]: an unsupervised shallow hashing method whose goals are to preserve locality and find balanced, uncorrelated hashes by solving the Eigenvector problem.
- *Spherical Hashing* (**SpheHash**) [14]: a hypersphere-based space-partitioning shallow hashing method.
- *Iterative Quantization* (**ITQ**) [10]: the state-of-the-art shallow hashing method that alternately minimizes the quantization error to achieve better hash codes.
- *Semantic Hashing* (**SemHash**) [26]: the widely-used unsupervised deep hashing method where the hash function is modeled as a deep generative model using Restricted Boltzmann Machine.
- *Self-taught Hashing* (**STH**) [38]: an extension of SemHash that also learns a hash function on unseen data.
- *Variational Deep Semantic Hashing* (**VDSH**) [5]: the deep hashing method that learns the hash function through a generative model using variational Autoencoders (VAE) with TF-IDF input.
- *Neural Architecture for Semantic Hashing* (**NASH**) [29]: the state-of-the-art deep hashing method that, similar to VDSH, learns the optimal hashing using VAE whose input are TF-IDF vectors. NASH models the hashing codes as a Bernoulli latent-variable.
- *Denoising Binary Autoencoder* (**DBA**): the binary autoencoder as defined in Section 3.2, with denoising input.
- *Denoising Adversarial Binary Autoencoder* (**DABA-MLP**): our proposed adversarially regularized autoencoder (described in Section 3.4) with multi-layer perceptrons as encoder/decoder whose input are the TF-IDF vectors of the documents.
- *DABA with RNN encoder/decoder* (**DABA-RNN**): our proposed DABA model (in Section 3.4) with RNN encoder and RNN decoder.
- *DABA with CNN encoder/decoder* (**DABA-CNN**): our proposed DABA model (in Section 3.4) with a Convolutional encoder and Deconvolutional decoder.

First, we pre-process the data by removing common stop words. Then, we transform each document into its TF-IDF [27] vector representations with 2,000 components for 20Newsgroup and Reuters and 5,000 components for DBpedia. For DABA-CNN and DABA-RNN, we learn the hash functions directly from the sequential text data.

We run the experiments of LSH using random projection on TF-IDF input. For SemHash and STH, we employ the network sizes similar to the ones used in the original papers [26, 38] for both the 20Newsgroup and Reuters datasets and two layers of 1000 $\rightarrow$ 500 units for DBpedia. We use similar network sizes for DABA-MLP. For 20Newsgroup and Reuters, VDSH's networks use 1,000 hidden nodes (as suggested in [5]), while NASH's networks have two layers of 500 $\rightarrow$ 500 units (as suggested in [29]); for DBpedia, 1,500 hidden nodes for VDSH and two layers of 1000 $\rightarrow$ 500 units for NASH. For DABA-RNN, we use an embedding size of 200 for 20Newsgroup and Reuters and 300 for DBpedia. For all DABA methods, the discriminator D is a MLP with two layers of 500 $\rightarrow$ 500 hidden units. To choose the denoising parameter p_d, we perform a grid search with p_d ranging from 0 to 0.5. For all these methods, we report the average MAP@1000 (shown as MAP) and Precision@k by repeating the experiments three times in order to ensure statistically meaningful results.

We implemented our proposed methods using Tensorflow[3]. In our experiments, we employ the learning rate of 0.001 for 20Newsgroup and Reuters datasets; 0.01 for the DBpedia dataset. We use a dropout rate of 0.2 and 0.5 for the small and large datasets, respectively. We also apply batch normalization [15] for our DABA-CNN method. Given N examples, the best method of solving the OT's LP program has a cost of approximately $O(N^{2.5}\log(N\mathcal{D}))$, where $\mathcal{D} = max_{i,j} d(b^{(i)}, z^{(j)})$ and the distances $d(b^{(i)}, z^{(j)})$ are scaled up to be integers [4]. While this is computationally expensive for large N, the cost is less than 100ms (where N is less than 2048) in a conventional CPU [4]. Therefore, it is entirely possible to implement this LP program in a mini-batch SGD training. In our experiments, we operate on a large mini-batch size of 500 examples in training

[3]https://www.tensorflow.org/
[4]https://github.com/gatagat/lap

DABA. This reduces the variance of the empirical primal Wasserstein estimate, while making the training process very efficient. The code of our proposed method is released on Github [5].

4.4 Results

4.4.1 Retrieval Performance. In this experiment, we measure the performance of the compared methods on the *document retrieval task*. Table 3 shows the average precision values at various retrieval thresholds m and the average MAP. Improvements of our models over the compared methods are statistically significant according to the corresponding paired t-tests (p-value < 0.01). As expected, shallow-hash functions have worse performance than other deep hashing methods. SemHash and DBA have comparable performance across the metric, which supports our discussion/claims in Section 3.3 that denoising autoencoders achieve a similar result as that of RBM-based autoencoders. However, RBM-based autoencoders are harder to train.

While there are mixed results when comparing the performance of the existing shallow hashing methods and the compared deep models, it is clearly observed that the proposed DABA-MLP significantly improves the retrieval quality compared to all these methods, including DBA – the DABA-MLP version that is not adversarially regularized. This supports our discussion/claims in Section 2 that adversarial training, besides generating better hash codes, helps the network learn a better semantic manifold of the data than that of a similar network but without adversarial regularization (this statement will again be supported by the qualitative experiments about the learned manifold of the data in Section 4.4.2). Finally, from Table 3, it is clearly noticed that DABA-RNN and DABA-CNN significantly outperform all other methods for most of the retrieval metrics. Also, DABA-CNN achieves better results compared to DABA-RNN, albeit comparable. We conjecture that the superior performance of all DABA methods is due to the following reasons: (1) the hash functions learned by DABA better preserve the semantic manifold of the documents (DABA-MLP performs better than DBA and SemHash/STH), (2) exploiting the semantic and syntactic structures of text documents (one of the main contributions of this paper) results in better retrieval quality (DABA-RNN/CNN perform better than DABA-MLP) and (3) the learned hash functions generate better hash codes (DABA-MLP performs better than all other shallow and deep models).

4.4.2 Hashing Space Manifold. In this experiment, we plot the embedding of the documents in the binary address space using the 2-D t-SNE projection [23], of the learned binary codes of various methods. In Figure 5 (created using the 20NewsGroups dataset) the documents belonging to the same topic are represented using the same color. As observed in this figure, in the DABA-RNN/CNN results, more documents belonging to the same topics are clustered together (while the clusters are more distant from each other) compared to the selected shallow and deep methods. Most "Religion/Christian" documents (purple) or "Sport/Baseball" documents (green), for example, are better separated from the rest of the documents. This demonstrates the effectiveness of the proposed method in preserving the semantic manifold of the documents in the coding space.

4.4.3 Analysis with Missing Data. In this experiment, we evaluate the performance, specifically Precision@100, of various selected methods against our proposed models when the data contains missing values. In the 20NewsGroups dataset, for each document, we randomly remove a predefined fraction of the words ranging from 0% to 100% in the query documents. We investigate the performance of our methods, DABA-CNN and DABA-RNN, when the denoising components are removed, resulting in ABA-CNN and ABA-RNN, respectively. This allows us to carry out an ablation study to understand the contribution of the proposed denoising component with regards to learning robust hash functions.

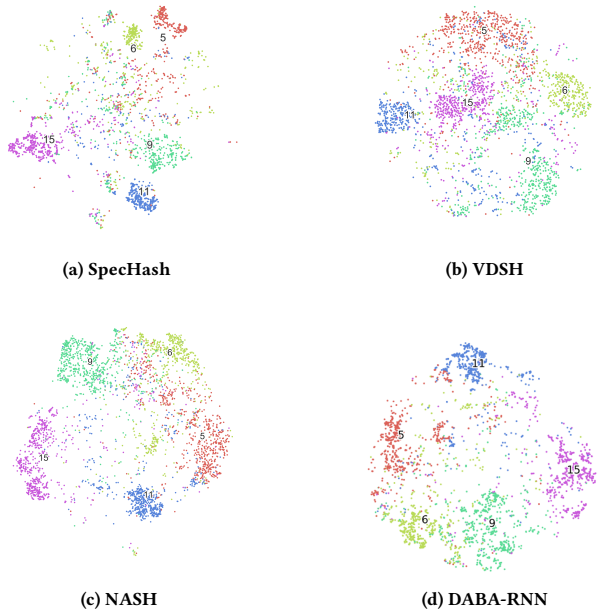
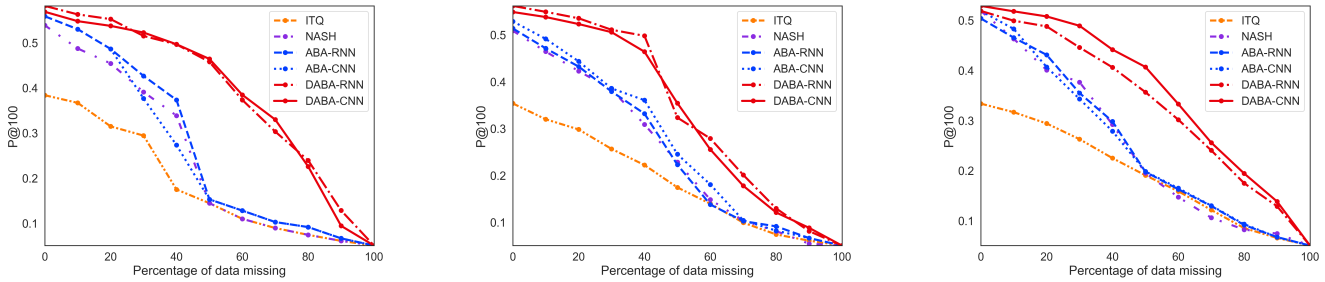


Figure 5: Two-dimensional t-SNE manifold visualization of the coding space (values of b) for five randomly selected topics, namely, Computer/Windows (5, Red), For Sale (6, Yellow-Green), Sport/Baseball (9, Green), Science/Cryptology (11, Blue), and Religion/Christian (15, Purple).

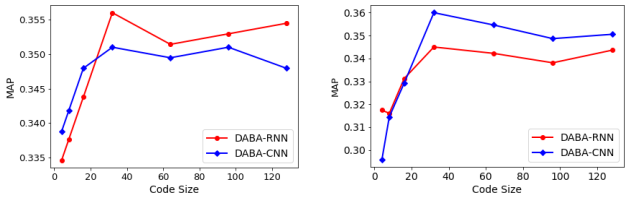
Figure 6 shows the performance of the methods at various percentages of missing data for different bit sizes on the 20Newsgroup dataset. The compared models, including ABA-CNN and ABA-RNN, have steeper performance decline than those of DABA-CNN and DABA-RNN when the missing data rate increases. DABA-CNN and DABA-RNN are more robust than the other methods in the presence of missing data. Note that all methods converge to the same performance, that is of the random guess, when the missing data rate is approaching 100%. The results demonstrate the robustness of employing the denoising component within our hashing framework toward missing data, a realistic scenario when deploying the similarity search system in a real-world environment.

[5] https://github.com/khoadoan/daba-hashing



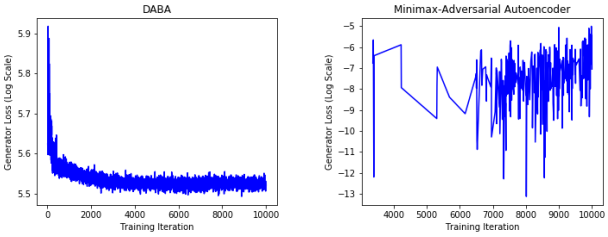
(a) 32-bit code (b) 64-bit code (c) 128-bit code

Figure 6: Precision@100 of various methods including DABA-CNN and DABA-RNN for various bit sizes on the 20Newsgroup dataset. ABA-RNN and ABA-CNN are our proposed methods without the denoising component.



(a) 20NewsGroups (c) DBpedia

Figure 7: Performance results (MAP) when varying the size of the learned binary codes on 20Newsgroup and DBpedia.



(a) DABA Training (b) Min-max Training

Figure 8: Convergence behavior of our proposed loss functions during the training process of 20Newsgroup. DABA is trained using the empirical OT loss without the min-max game. The min-max autoencoder is trained using the Wasserstein Kantorovich-Rubinstein dual. In this game, the loss has poor fluctuation, thus is it difficult to know when to stop training.

4.4.4 Parameter Sensitivity. In this experiment, we study the relationship between the embedding size B of the binary code b and the retrieval performance. Figure 7 illustrates the MAP performance of DABA-CNN as the value of B is varied from 4 to 128. We observe that the retrieval performance quickly increases during the initial phase (when using smaller codes), following which it reaches the optimal value. Increasing the code sizes after that optimal code size results in neither performance gain nor any significant performance loss. This suggests that if we train DABA networks with a sufficiently large code size B, we are guaranteed to obtain a near-optimal performance. This experiment illustrates that the performance of the proposed DABA model is not sensitive to the learned code size parameter.

4.4.5 Stability during training. One of the subtleties of training a min-max adversarial network is to decide the stopping criterion for the training process, that is "when does the model reach the equilibrium?". Figure 8 illustrates the generator's loss, L_M, for Wasserstein-Autoencoder [31] and DABA in the 20Newsgroup training. We observe that the loss fluctuates in the min-max game during training, while our proposed OT loss gradually descends. This displays the training efficiency of our proposed training algorithm. This effectively suggests that we can utilize the criteria to detect convergence such as early stopping to automatically train DABA, making it suitable for real-world hashing problems.

5 CONCLUSION

We proposed a novel and efficient denoising adversarial binary autoencoder for the text hashing problem. To achieve this, we developed a new and effective mechanism to constrain the network's hidden layer and force them to become binary by regularizing the hash-function learning process with an adversarial network with a discrete distribution prior and corrupted input that the denoising autoencoder must reconstruct. Employing the OT formulation, we train our proposed model without the min-max game and significantly stabilize the training process of the adversarial autoencoder. We demonstrate that the proposed model using RNN or CNN encoder/decoder, that effectively exploits the semantic and syntactic dependencies of text documents, outperforms other existing state-of-the-art semantic-similarity search methods.

ACKNOWLEDGMENTS

This work was supported in part by the US National Science Foundation grants IIS-1619028, IIS-1707498 and IIS-1838730.

REFERENCES

[1] Martin Arjovsky, Soumith Chintala, and Léon Bottou. 2017. Wasserstein gan. *arXiv preprint arXiv:1701.07875* (2017).

[2] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. 2013. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*. 899–907.

[3] David Berthelot, Colin Raffel, Aurko Roy, and Ian Goodfellow. 2018. Understanding and improving interpolation in autoencoders via an adversarial regularizer. *arXiv preprint arXiv:1807.07543* (2018).

[4] Rainer E Burkard, Mauro Dell'Amico, and Silvano Martello. 2009. *Assignment problems*. Springer.

[5] Suthee Chaidaroon and Yi Fang. 2017. Variational deep semantic hashing for text documents. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 75–84.

[6] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 1724–1734.

[7] Khoa D Doan, Pranjul Yadav, and Chandan K Reddy. 2019. Adversarial Factorization Autoencoder for Look-alike Modeling. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 2803–2812.

[8] Wei Dong, Zhe Wang, William Josephson, Moses Charikar, and Kai Li. 2008. Modeling LSH for performance tuning. In *Proceedings of the 17th ACM conference on Information and knowledge management*. ACM, 669–678.

[9] Tiezheng Ge, Kaiming He, and Jian Sun. 2014. Graph cuts for supervised binary coding. In *European Conference on Computer Vision*. Springer, 250–264.

[10] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35, 12 (2013), 2916–2929.

[11] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep learning*. MIT press.

[12] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. 2672–2680.

[13] Junfeng He, Shih-Fu Chang, Regunathan Radhakrishnan, and Claus Bauer. 2011. Compact hashing with joint optimization of search accuracy and time. In *CVPR 2011*. IEEE, 753–760.

[14] Jae-Pil Heo, Youngwoon Lee, Junfeng He, Shih-Fu Chang, and Sung-Eui Yoon. 2012. Spherical hashing. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2957–2964.

[15] Sergey Ioffe and Christian Szegedy. 2015. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *International Conference on Machine Learning*. 448–456.

[16] Alexis Joly and Olivier Buisson. 2011. Random maximum margin hashing. In *CVPR 2011*. IEEE, 873–880.

[17] Yoon Kim, Kelly Zhang, Alexander M Rush, Yann LeCun, et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223* (2017).

[18] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[19] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[20] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*. 3294–3302.

[21] Jure Leskovec, Anand Rajaraman, and Jeffrey David Ullman. 2014. *Mining of massive datasets*. Cambridge university press.

[22] Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*. IEEE, 2074–2081.

[23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, Nov (2008), 2579–2605.

[24] Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2015. Adversarial autoencoders. *arXiv preprint arXiv:1511.05644* (2015).

[25] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Eleventh Annual Conference of the International Speech Communication Association*.

[26] Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *International Journal of Approximate Reasoning* 50, 7 (2009), 969–978.

[27] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[28] Gunnar Schröder, Maik Thiele, and Wolfgang Lehner. 2011. Setting goals and choosing metrics for recommender system evaluations. In *UCERSTI2 Workshop at the 5th ACM Conference on Recommender Systems, Chicago, USA*, Vol. 23. 53.

[29] Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Ricardo Henao, and Lawrence Carin. 2018. NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2041–2050.

[30] Malcolm Slaney, Yury Lifshits, and Junfeng He. 2012. Optimal parameters for locality-sensitive hashing. *Proc. IEEE* 100, 9 (2012), 2604–2623.

[31] Ilya Tolstikhin, Olivier Bousquet, Sylvain Gelly, and Bernhard Schoelkopf. 2017. Wasserstein auto-encoders. *arXiv preprint arXiv:1711.01558* (2017).

[32] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.

[33] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*. ACM, 1096–1103.

[34] Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for similarity search: A survey. *arXiv preprint arXiv:1408.2927* (2014).

[35] Jingdong Wang, Ting Zhang, Nicu Sebe, Heng Tao Shen, et al. 2017. A survey on learning to hash. *IEEE transactions on pattern analysis and machine intelligence* (2017).

[36] Qifan Wang, Dan Zhang, and Luo Si. 2013. Semantic hashing using tags and topic modeling. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*. ACM, 213–222.

[37] Yair Weiss, Antonio Torralba, and Rob Fergus. 2009. Spectral hashing. In *Advances in neural information processing systems*. 1753–1760.

[38] D Zhang, J Wang, D Cai, and J Lu. 2010. Self-taught hashing for fast similarity search. In *SIGIR'10: Proceeding of the 33rd international ACM SIGIR conference on Research and development in information retrieval*. ACM, 18–25.

[39] Xiang Zhang and Yann LeCun. 2015. Text understanding from scratch. *arXiv preprint arXiv:1502.01710* (2015).

[40] Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*. 649–657.

[41] Yuyu Zhang, Hanjun Dai, Chang Xu, Jun Feng, Taifeng Wang, Jiang Bian, Bin Wang, and Tie-Yan Liu. 2014. Sequential Click Prediction for Sponsored Search with Recurrent Neural Networks.. In *AAAI*. 1369–1375.

[42] Yizhe Zhang, Dinghan Shen, Guoyin Wang, Zhe Gan, Ricardo Henao, and Lawrence Carin. 2017. Deconvolutional paragraph representation learning. In *Advances in Neural Information Processing Systems*. 4169–4179.