Abstract

Semantic hashing has become a powerful paradigm for fast similarity search in many information retrieval systems. While fairly successful, previous techniques generally require two-stage training, and the binary constraints are handled ad-hoc. In this paper, we present an end-to-end Neural Architecture for Semantic Hashing (NASH), where the binary hashing codes are treated as Bernoulli latent variables. A neural variational inference framework is proposed for training, where gradients are directly backpropagated through the discrete latent variable to optimize the hash function. We also draw connections between proposed method and rate-distortion theory, which provides a theoretical foundation for the effectiveness of the proposed framework. Experimental results on three public datasets demonstrate that our method significantly outperforms several state-of-the-art models on both unsupervised and supervised scenarios.

1 Introduction

The problem of similarity search, also called nearest-neighbor search, consists of finding documents from a large collection of documents, or corpus, which are most similar to a query document of interest. Fast and accurate similarity search is at the core of many information retrieval applications, such as plagiarism analysis (Stein et al., 2007), collaborative filtering (Koren, 2008), content-based multimedia retrieval (Lew et al., 2006) and caching (Pandey et al., 2009). Semantic hashing is an effective approach for fast similarity search (Salakhutdinov and Hinton, 2009; Zhang et al., 2010; Wang et al., 2014). By representing every document in the corpus as a similarity-preserving discrete (binary) hashing code, the similarity between two documents can be evaluated by simply calculating pairwise Hamming distances between hashing codes, i.e., the number of bits that are different between two codes. Given that today, an ordinary PC is able to execute millions of Hamming distance computations in just a few milliseconds (Zhang et al., 2010), this semantic hashing strategy is very computationally attractive.

While considerable research has been devoted to text (semantic) hashing, existing approaches typically require two-stage training procedures. These methods can be generally divided into two categories: (i) binary codes for documents are first learned in an unsupervised manner, then l-bit binary classifiers are trained via supervised learning to predict the l-bit hashing code (Zhang et al., 2010; Xu et al., 2015); (ii) continuous text representations are first inferred, which are binarized as a second (separate) step during testing (Wang et al., 2013; Chaidaroon and Fang, 2017). Because the model parameters are not learned in an end-to-end manner, these two-stage training strategies may result in suboptimal local optima. This happens because different modules within the model are optimized separately, preventing the sharing of information between them. Further, in existing methods, binary constraints are typically handled ad-hoc by truncation, i.e., the hashing codes are obtained via direct binarization from continuous representations after training. As a result, the information contained in the continuous representations is lost during the (separate) binarization process. Moreover, training different modules (mapping and classifier/binarization) separately often requires additional hyperparameter tuning for each training stage, which can be laborious and time-consuming.
In this paper, we propose a simple and generic neural architecture for text hashing that learns binary latent codes for documents in an end-to-end manner. Inspired by recent advances in neural variational inference (NVI) for text processing (Miao et al., 2016; Yang et al., 2017; Shen et al., 2017b), we approach semantic hashing from a generative model perspective, where binary (hashing) codes are represented as either deterministic or stochastic Bernoulli latent variables. The inference (encoder) and generative (decoder) networks are optimized jointly by maximizing a variational lower bound to the marginal distribution of input documents (corpus). By leveraging a simple and effective method to estimate the gradients with respect to discrete (binary) variables, the loss term from the generative (decoder) network can be directly backpropagated into the inference (encoder) network to optimize the hash function.

Motivated by the rate-distortion theory (Berger, 1971; Theis et al., 2017), we propose to inject data-dependent noise into the latent codes during the decoding stage, which adaptively accounts for the tradeoff between minimizing rate (number of bits used, or effective code length) and distortion (reconstruction error) during training. The connection between the proposed method and rate-distortion theory is further elucidated, providing a theoretical foundation for the effectiveness of our framework.

Summarizing, the contributions of this paper are: (i) to the best of our knowledge, we present the first semantic hashing architecture that can be trained in an end-to-end manner; (ii) we propose a neural variational inference framework to learn compact (regularized) binary codes for documents, achieving promising results on both unsupervised and supervised text hashing; (iii) the connection between our method and rate-distortion theory is established, from which we demonstrate the advantage of injecting data-dependent noise into the latent variable during training.

2 Related Work

Models with discrete random variables have attracted much attention in the deep learning community (Jang et al., 2016; Maddison et al., 2016; van den Oord et al., 2017; Li et al., 2017; Shu and Nakayama, 2017). Some of these structures are more natural choices for language or speech data, which are inherently discrete. More specifically, van den Oord et al. (2017) combined VAEs with vector quantization to learn discrete latent representation, and demonstrated the utility of these learned representations on images, videos, and speech data. Li et al. (2017) leveraged both pairwise label and classification information to learn discrete hash codes, which exhibit state-of-the-art performance on image retrieval tasks.

For natural language processing (NLP), although significant research has been made to learn continuous deep representations for words or documents (Mikolov et al., 2013; Kiros et al., 2015; Shen et al., 2018), discrete neural representations have been mainly explored in learning word embeddings (Shu and Nakayama, 2017; Chen et al., 2017). In these recent works, words are represented as a vector of discrete numbers, which are very efficient storage-wise, while showing comparable performance on several NLP tasks, relative to continuous word embeddings. However, discrete representations that are learned in an end-to-end manner at the sentence or document level have been rarely explored. Also there is a lack of strict evaluation regarding their effectiveness. Our work focuses on learning discrete (binary) representations for text documents. Further, we employ semantic hashing (fast similarity search) as a mechanism to evaluate the quality of learned binary latent codes.

3 The Proposed Method

3.1 Hashing under the NVI Framework

Inspired by the recent success of variational autoencoders for various NLP problems (Miao et al., 2016; Bowman et al., 2015; Yang et al., 2017; Miao et al., 2017; Shen et al., 2017b; Wang et al., 2018), we approach the training of discrete (binary) latent variables from a generative perspec-
tive. Let $x$ and $z$ denote the input document and its corresponding binary hash code, respectively. Most of the previous text hashing methods focus on modeling the encoding distribution $p(z|x)$, or hash function, so the local/global pairwise similarity structure of documents in the original space is preserved in latent space (Zhang et al., 2010; Wang et al., 2013; Xu et al., 2015; Wang et al., 2014). However, the generative (decoding) process of reconstructing $x$ from binary latent code $z$, i.e., modeling distribution $p(x|z)$, has been rarely considered. Intuitively, latent codes learned from a model that accounts for the generative term should naturally encapsulate key semantic information from $x$ because the generation/reconstruction objective is a function of $p(x|z)$. In this regard, the generative term provides a natural training objective for semantic hashing.

We define a generative model that simultaneously accounts for both the encoding distribution, $p(z|x)$, and decoding distribution, $p(x|z)$, by defining approximations $q_\theta(z|x)$ and $q_\phi(x|z)$, via inference and generative networks, $g_\phi(x)$ and $g_\theta(z)$, parameterized by $\phi$ and $\theta$, respectively. Specifically, $x \in \mathbb{Z}_H^{|V|}$ is the bag-of-words (count) representation for the input document, where $|V|$ is the vocabulary size. Notably, we can also employ other count weighting schemes as input features $x$, e.g., the term frequency-inverse document frequency (TFIDF) (Manning et al., 2008). For the encoding distribution, a latent variable $z$ is first inferred from the input text $x$, by constructing an inference network $g_\phi(x)$ to approximate the true posterior distribution $p(z|x)$ as $q_\phi(z|x)$. Subsequently, the decoder network $g_\theta(z)$ maps $z$ back into input space to reconstruct the original sequence $x$ as $\hat{x}$, approximating $p(x|z)$ as $q_\theta(x|z)$ (as shown in Figure 1). This cyclic strategy, $x \rightarrow z \rightarrow \hat{x} \approx x$, provides the latent variable $z$ with a better ability to generalize (Miao et al., 2016).

To tailor the NVI framework for semantic hashing, we cast $z$ as a binary latent variable and assume a multivariate Bernoulli prior on $z$: $p(z) \sim$ Bernoulli($\gamma$) = $\prod_{i=1}^{n} \gamma_i^{z_i}(1 - \gamma_i)^{1-z_i}$, where $\gamma_i \in [0, 1]$ is component $i$ of vector $\gamma$. Thus, the encoding (approximate posterior) distribution $q_\phi(z|x)$ is restricted to take the form $q_\phi(z|x) = \text{Bernoulli}(h)$, where $h = \sigma(g_\phi(x))$, $\sigma(\cdot)$ is the sigmoid function, and $g_\phi(\cdot)$ is the (nonlinear) inference network specified as a multilayer perceptron (MLP). As illustrated in Figure 1, we can obtain samples from the Bernoulli posterior either deterministically or stochastically. Suppose $z$ is a $l$-bit hash code, for the deterministic binarization, we have, for $i = 1, 2, \ldots, l$:

$$z_i = 1_{\sigma(g_\phi(x)) > 0.5} = \frac{\text{sign}(\sigma(g_\phi(x) - 0.5) + 1}{2},$$

where $z$ is the binarized variable, and $z_i$ and $g_\phi(x)$ denote the $i$-th dimension of $z$ and $g_\phi(x)$, respectively. The standard Bernoulli sampling in (1) can be understood as setting a hard threshold at 0.5 for each representation dimension, therefore, the binary latent code is generated deterministically. Another strategy to obtain the discrete variable is to binarize $h$ in a stochastic manner:

$$z_i = 1_{\sigma(g_\phi(x)) > \mu_i} = \frac{\text{sign}(\sigma(g_\phi(x) - \mu_i) + 1}{2},$$

where $\mu_i \sim \text{Uniform}(0, 1)$. Because of this sampling process, we do not have to assume a pre-defined threshold value like in (1).

### 3.2 Training with Binary Latent Variables

To estimate the parameters of the encoder and decoder networks, we would ideally maximize the marginal distribution $p(x) = \int p(z)p(x|z)dz$. However, computing this marginal is intractable in most cases of interest. Instead, we maximize a variational lower bound. This approach is typically employed in the VAE framework (Kingma and Welling, 2013):

$$L_{\text{vae}} = \mathbb{E}_{q_\phi(z|x)} \left[ \log q_\theta(x|z)p(z) q_\phi(z|x) \right],$$

$$= \mathbb{E}_{q_\phi(z|x)} \left[ \log q_\theta(x|z) - D_{KL}(q_\phi(z|x)||p(z)) \right],$$

where the Kullback-Leibler (KL) divergence $D_{KL}(q_\phi(z|x)||p(z))$ encourages the approximate posterior distribution $q_\phi(z|x)$ to be close to the multivariate Bernoulli prior $p(z)$. In this case, $D_{KL}(q_\phi(z|x)||p(z))$ can be written in closed-form as a function of $g_\phi(x)$:

$$D_{KL} = g_\phi(x) \log \frac{g_\phi(x)}{\gamma} + (1 - g_\phi(x)) \log \frac{1 - g_\phi(x)}{1 - \gamma}.$$
For the first term in (3), we should in principle estimate the influence of $\mu_i$ in (2) on $q_\theta(x|z)$ by averaging over the entire (uniform) noise distribution. However, a closed-form distribution does not exist since it is not possible to enumerate all possible configurations of $z$, especially when the latent dimension is large. Moreover, discrete latent variables are inherently incompatible with backpropagation, since the derivative of the sign function is zero for almost all input values. As a result, the exact gradients of $L_{vae}$ wrt the inputs before binarization would be essentially all zero.

To estimate the gradients for binary latent variables, we utilize the straight-through (ST) estimator, which was first introduced by Hinton (2012). So motivated, the strategy here is to simply backpropagate through the hard threshold by approximating the gradient $\partial z / \partial \phi$ as 1. Thus, we have:

$$
\frac{dE_{q_\theta(z|x)}[\log q_\theta(x|z)]}{d\phi} = \frac{dE_{q_\theta(z|x)}[\log q_\theta(x|z)]}{dz} \frac{dz}{d\sigma(g_{\phi}^i(x))} \frac{d\sigma(g_{\phi}^i(x))}{d\phi}
$$

(5)

Although this is clearly a biased estimator, it has been shown to be a fast and efficient method relative to other gradient estimators for discrete variables, especially for the Bernoulli case (Bengio et al., 2013; Hubara et al., 2016; Theis et al., 2017). With the ST gradient estimator, the first loss term in (3) can be backpropagated into the encoder network to fine-tune the hash function $g_{\phi}(x)$.

For the approximate generator $q_\theta(x|z)$ in (3), let $x_i$ denote the one-hot representation of $i$th word within a document. Note that $x = \sum_i x_i$ is thus the bag-of-words representation for document $x$. To reconstruct the input $x$ from $z$, we utilize a softmax decoding function written as:

$$
q(x_i = w | z) = \frac{\exp(z^T E_{x_w} + b_w)}{\sum_{j=1}^{|V|} \exp(z^T E_{x_j} + b_j)},
$$

(6)

where $q(x_i = w | z)$ is the probability that $x_i$ is word $w \in V$, $q_\theta(x|z) = \prod_i q(x_i = w | z)$ and $\theta = \{E, b_1, \ldots, b_{|V|}\}$. Note that $E \in \mathbb{R}^{d \times |V|}$ can be interpreted as a word embedding matrix to be learned, and $\{b_i\}_{i=1}^{|V|}$ denote bias terms. Intuitively, the objective in (6) encourages the discrete vector $z$ to be close to the embeddings for every word that appear in the input document $x$. As shown in Section 5.3.1, meaningful semantic structures can be learned and manifested in the word embedding matrix $E$.

### 3.3 Injecting Data-dependent Noise to $z$

To reconstruct text data $x$ from sampled binary representation $z$, a deterministic decoder is typically utilized (Miao et al., 2016; Chaidaroon and Fang, 2017). Inspired by the success of employing stochastic decoders in image hashing applications (Dai et al., 2017; Theis et al., 2017), in our experiments, we found that injecting random Gaussian noise into $z$ makes the decoder a more favorable regularizer for the binary codes, which in practice leads to stronger retrieval performance. Below, we invoke the rate-distortion theory to perform some further analysis, which leads to interesting findings.

Learning binary latent codes $z$ to represent a continuous distribution $p(x)$ is a classical information theory concept known as lossy source coding. From this perspective, semantic hashing, which compresses an input document into compact binary codes, can be casted as a conventional rate-distortion tradeoff problem (Theis et al., 2017; Ballé et al., 2016):

$$
\min \text{Rate} - \log_2 R(z) + \beta \cdot \text{Distortion} = D(x, \hat{x}),
$$

(7)

where rate and distortion denote the effective code length, i.e., the number of bits used, and the distortion introduced by the encoding/decoding sequence, respectively. Further, $\hat{x}$ is the reconstructed input and $\beta$ is a hyperparameter that controls the tradeoff between the two terms.

Considering the case where we have a Bernoulli prior on $z$ as $p(z) \sim \text{Bernoulli}(\gamma)$, and $x$ conditionally drawn from a Gaussian distribution $p(x|z) \sim \mathcal{N}(Ez, \sigma^2 I)$. Here, $E = \{e_i\}_{i=1}^{|V|}$, where $e_i \in \mathbb{R}^d$ can be interpreted as a codebook with $|V|$ codewords. In our case, $E$ corresponds to the word embedding matrix as in (6).

For the case of stochastic latent variable $z$, the objective function in (3) can be written in a form similar to the rate-distortion tradeoff:

$$
\min \mathbb{E}_{q_\theta(z|x)} \left[ -\log q_\theta(z|x) + \frac{1}{2\sigma^2} ||x - Ez||^2 + C \right],
$$

(8)
where $C$ is a constant that encapsulates the prior distribution $p(z)$ and the Gaussian distribution normalization term. Notably, the trade-off hyper-parameter $\beta = \sigma^{-2}/2$ is closely related to the variance of the distribution $p(x|z)$. In other words, by controlling the variance $\sigma$, the model can adaptively explore different trade-offs between the rate and distortion objectives. However, the optimal trade-offs for distinct samples may be different.

Inspired by the observations above, we propose to inject data-dependent noise into latent variable $z$, rather than to setting the variance term $\sigma^2$ to a fixed value (Dai et al., 2017; Theis et al., 2017). Specifically, $\log \sigma^2$ is obtained via a one-layer MLP transformation from $g_\phi(x)$. Afterwards, we sample $z'$ from $\mathcal{N}(z, \sigma^2 I)$, which then replace $z$ in (6) to infer the probability of generating individual words (as shown in Figure 1). As a result, the variances are different for every input document $x$, and thus the model is provided with additional flexibility to explore various trade-offs between rate and distortion for different training observations. Although our decoder is not a strictly Gaussian distribution, as in (6), we found empirically that injecting data-dependent noise into $z$ yields strong retrieval results, see Section 5.1.

### 3.4 Supervised Hashing

The proposed Neural Architecture for Semantic Hashing (NASH) can be extended to supervised hashing, where a mapping from latent variable $z$ to labels $y$ is learned, here parametrized by a two-layer MLP followed by a fully-connected softmax layer. To allow the model to explore and balance the model is provided with additional flexibility to explore various trade-offs between rate and distortion for different training observations. Although our decoder is not a strictly Gaussian distribution, as in (6), we found empirically that injecting data-dependent noise into $z$ yields strong retrieval results, see Section 5.1.

\[
L = -L_{\text{vae}}(\theta, \phi; x) + \alpha L_{\text{dis}}(\eta; z, y).
\]

where $\eta$ refers to parameters of the MLP classifier and $\alpha$ controls the relative weight between the variational lower bound ($L_{\text{vae}}$) and discriminative loss ($L_{\text{dis}}$), defined as the cross-entropy loss. The parameters $\{\theta, \phi, \eta\}$ are learned end-to-end via Monte Carlo estimation.

### 4 Experimental Setup

#### 4.1 Datasets

We use the following three standard publicly available datasets for training and evaluation:

1. **Reuters21578**, containing 10,788 news documents, which have been classified into 90 different categories.
2. **20NewsGroups**, a collection of 18,828 newsgroup documents, which are categorized into 20 different topics.
3. **TMC** (stands for SIAM text mining competition), containing air traffic reports provided by NASA. TMC consists of 21,519 training documents divided into 22 different categories. To make direct comparison with prior works, we employed the TFIDF features on these datasets supplied by (Chaidaroon and Fang, 2017), where the vocabulary sizes for these three datasets are set to 10,000, 7,164 and 20,000, respectively.

#### 4.2 Training Details

For the inference networks, we employ a feed-forward neural network with 2 hidden layers (both with 500 units) using the ReLU non-linearity activation function, which transform the input documents, i.e., TFIDF features in our experiments, into a continuous representation. Empirically, we found that stochastic binarization as in (2) shows stronger performance than deterministic binarization, and thus use the former in our experiments. However, we further conduct a systematic ablation study in Section 5.2 to compare the two binarization strategies.

Our model is trained using Adam (Kingma and Ba, 2014), with a learning rate of $1 \times 10^{-3}$ for all parameters. We decay the learning rate by a factor of 0.96 for every 10,000 iterations. Dropout (Srivastava et al., 2014) is employed on the output of encoder networks, with the rate selected from $\{0.7, 0.8, 0.9\}$ on the validation set. To facilitate comparisons with previous methods, we set the dimension of $z$, i.e., the number of bits within the hashing code) as 8, 16, 32, 64, or 128.

#### 4.3 Baselines

We evaluate the effectiveness of our framework on both unsupervised and supervised semantic hashing tasks. We consider the following unsupervised baselines for comparisons: Locality Sensitive Hashing (LSH) (Datar et al., 2004), Stack Restricted Boltzmann Machines (S-RBM) (Salakhutdinov and Hinton, 2009), Spectral Hashing (SpH) (Weiss et al., 2009), Self-taught Hashing (STH) (Zhang et al., 2010) and Variational Deep Semantic Hashing (VDSH) (Chaidaroon and Fang, 2017).
Table 1: Precision of the top 100 retrieved documents on Reuters dataset (Unsupervised hashing).

| Method      | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits |
|-------------|--------|---------|---------|---------|----------|
| LSH         | 0.2802 | 0.3215  | 0.3862  | 0.4667  | 0.5194   |
| S-RBM       | 0.5113 | 0.5740  | 0.6154  | 0.6177  | 0.6452   |
| Sph         | 0.6080 | 0.6340  | 0.6513  | 0.6290  | 0.6045   |
| STH         | 0.6616 | 0.7351  | 0.7554  | 0.7350  | 0.6986   |
| VDSH        | 0.6859 | 0.7165  | 0.7753  | 0.7456  | 0.7318   |
| NASH        | 0.7113 | 0.7624  | 0.7993  | 0.7812  | 0.7559   |
| NASH-N      | 0.7352 | 0.7904  | 0.8297  | 0.8086  | 0.7867   |
| NASH-DN     | 0.7470 | 0.8013  | 0.8418  | 0.8297  | 0.7924   |

Figure 2: Precision of the top 100 retrieved documents on Reuters dataset (Supervised hashing), compared with other supervised baselines.

5.1 Semantic Hashing Evaluation

Table 1 presents the results of all models on Reuters dataset. Regarding unsupervised semantic hashing, all the NASH variants consistently outperform the baseline methods by a substantial margin, indicating that our model makes the most effective use of unlabeled data and manages to assign similar hashing codes, i.e., with small Hamming distance to each other, to documents that belong to the same label. It can be also observed that the injection of noise into the decoder networks has improved the robustness of learned binary representations, resulting in better retrieval performance. More importantly, by making the variances of noise adaptive to the specific input, our NASH-DN achieves even better results, compared with NASH-N, highlighting the importance of exploring/learning the trade-off between rate and distortion objectives by the data itself. We observe the same trend and superiority of our NASH-DN models on the other two benchmarks, as shown in Tables 3 and 4.

Another observation is that the retrieval results tend to drop a bit when we set the length of hashing codes to be 64 or larger, which also happens for some baseline models. This phenomenon has been reported previously in Wang et al. (2012); Liu et al. (2012); Wang et al. (2013); Chaidaroon and Fang (2017), and the reasons could be twofold: (i) for longer codes, the number of data points that are assigned to a certain binary code decreases exponentially. As a result, many queries may fail to return any neighbor documents (Wang et al., 2012); (ii) considering the size of training data, it is likely that the model may overfit with long hash codes (Chaidaroon and Fang, 2017). However, even with longer hashing codes,
In this regard, we further investigate the effect of stochastic sampling in our NASH models perform stronger than the baselines in most cases (except for the 20Newsgroups dataset), suggesting that NASH can effectively allocate documents to informative/meaningful hashing codes even with limited training data.

We also evaluate the effectiveness of NASH in a supervised scenario on the Reuters dataset, where the label or topic information is utilized during training. As shown in Figure 2, our NASH-DN-S model consistently outperforms several supervised semantic hashing baselines, with various choices of hashing bits. Notably, our model exhibits higher Top-100 retrieval precision than VDSH-S and VDSH-SP, proposed by Chaidaroon and Fang (2017). This may be attributed to the fact that in VDSH models, the continuous embeddings are not optimized with their future binarization in mind, and thus could hurt the relevance of learned binary codes. On the contrary, our model is optimized in an end-to-end manner, where the gradients are directly backpropagated to the inference network (through the binary/discrete latent variable), and thus gives rise to a more robust hash function.

### 5.2 Ablation study

#### 5.2.1 The effect of stochastic sampling

As described in Section 3, the binary latent variables $z$ in NASH can be either deterministically (via (1)) or stochastically (via (2)) sampled. We compare these two types of binarization functions in the case of unsupervised hashing. As illustrated in Figure 3, stochastic sampling shows stronger retrieval results on all three datasets, indicating that endowing the sampling process of latent variables with more stochasticity improves the learned representations.

#### 5.2.2 The effect of encoder/decoder networks

Under the variational framework introduced here, the encoder network, i.e., hash function, and decoder network are jointly optimized to abstract semantic features from documents. An interesting question concerns what types of network should be leveraged for each part of our NASH model. In this regard, we further investigate the effect of

---

### Table 2: The five nearest words in the semantic space learned by NASH, compared with the results from NVDM (Miao et al., 2016).

| Method     | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits |
|------------|--------|---------|---------|---------|----------|
| NASH       |        |         |         |         |          |
| Weapons    | 0.7438 | 0.7012 | 0.6563 | 0.6103 | 0.5645   |
| Guns       | 0.6862 | 0.6451 | 0.5999 | 0.5559 | 0.5128   |
| Armed      | 0.6771 | 0.6360 | 0.5919 | 0.5488 | 0.5067   |
| Assault    | 0.6712 | 0.6299 | 0.5857 | 0.5428 | 0.5006   |
| NASH-DN    | 0.6586 | 0.6175 | 0.5724 | 0.5293 | 0.4872   |
| NVDM       | 0.6052 | 0.5541 | 0.5091 | 0.4660 | 0.4239   |
| Weapons    | 0.7438 | 0.7012 | 0.6563 | 0.6103 | 0.5645   |
| Guns       | 0.6862 | 0.6451 | 0.5999 | 0.5559 | 0.5128   |
| Armed      | 0.6771 | 0.6360 | 0.5919 | 0.5488 | 0.5067   |
| Assault    | 0.6712 | 0.6299 | 0.5857 | 0.5428 | 0.5006   |
| NVDM       | 0.6052 | 0.5541 | 0.5091 | 0.4660 | 0.4239   |

### Table 3: Precision of the top 100 retrieved documents on 20Newsgroups dataset.

| Method     | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits |
|------------|--------|---------|---------|---------|----------|
| LSH        | 0.4388 | 0.4393 | 0.4514 | 0.4553 | 0.4773   |
| S-RBM      | 0.4846 | 0.5108 | 0.5166 | 0.5190 | 0.5137   |
| Sph        | 0.2690 | 0.3235 | 0.2357 | 0.1411 | 0.1299   |
| STH        | 0.3723 | 0.3947 | 0.4105 | 0.4181 | 0.4123   |
| VDSH-S     | 0.4330 | 0.6853 | 0.7108 | 0.4410 | 0.5847   |
| VDSH-SP    | 0.5849 | 0.6573 | 0.6921 | 0.6548 | 0.5998   |
| NASH-N     | 0.6233 | 0.6759 | 0.7201 | 0.6877 | 0.6314   |
| NASH-DN    | 0.6358 | 0.6956 | 0.7327 | 0.7010 | 0.6325   |

### Table 4: Precision of the top 100 retrieved documents on TMC dataset.

| Method     | 8 bits | 16 bits | 32 bits | 64 bits | 128 bits |
|------------|--------|---------|---------|---------|----------|
| KSH        | 0.6060 | 0.6842 | 0.7047 | 0.7175 | 0.7243   |
| SHTM       | 0.6299 | 0.6571 | 0.6485 | 0.6893 | 0.6474   |
| VDSH-S     | 0.7387 | 0.7887 | 0.7897 | 0.7967 | 0.8018   |
| VDSH-SP    | 0.7498 | 0.7798 | 0.7891 | 0.7888 | 0.7970   |
| NASH-DN    | 0.7438 | 0.7946 | 0.7987 | 0.8014 | 0.8139   |
Category | Title/Subject | 8-bit code | 16-bit code
---|---|---|---
Baseball | Dave Kingman for the hall of fame | 11101001 | 0010101000000110
Time of game | 11111001 | 0010100100000111
Game score report | 11101001 | 0010110100000111
Why is Barry Bonds not batting 4th? | 11101101 | 00111101000000110
Electronics | Building a UV flashlight | 10110100 | 0010001000101011
How to drive an array of LEDs | 10110101 | 0010001000101001
2% silver solder | 11010101 | 0010001000101011
Subliminal message flashing on TV | 10110100 | 0010011000101001

Table 5: Examples of learned compact hashing codes on 20Newsgroups dataset.

Figure 3: The precisions of the top 100 retrieved documents for NASH-DN with stochastic or deterministic binary latent variables.

| Network      | Encoder | Decoder |
|--------------|---------|---------|
| linear       | 0.5844  | 0.6225  |
| one-layer MLP| 0.6187  | 0.3559  |
| two-layer MLP| 0.6225  | 0.1047  |

Table 6: Ablation study with different encoder/decoder networks.

It is observed that for the encoder networks, increasing the non-linearity by stacking MLP layers leads to better empirical results. In other words, endowing the hash function with more modeling capacity is advantageous to retrieval tasks. However, when we employ a non-linear network for the decoder, the retrieval precision drops dramatically. It is worth noting that the only difference between linear transformation and one-layer MLP is whether a non-linear activation function is employed or not.

This observation may be attributed the fact that the decoder networks can be considered as a similarity measure between latent variable $z$ and the word embeddings $E_k$ for every word, and the probabilities for words that present in the document is maximized to ensure that $z$ is informative. As a result, if we allow the decoder to be too expressive (e.g., a one-layer MLP), it is likely that we will end up with a very flexible similarity measure but relatively less meaningful binary representations. This finding is consistent with several image hashing methods, such as SGH (Dai et al., 2017) or binary autoencoder (Carreira-Perpinán and Raziperchikolaei, 2015), where a linear decoder is typically adopted to obtain promising retrieval results. However, our experiments may not speak for other choices of encoder-decoder architectures, e.g., LSTM-based sequence-to-sequence models (Sutskever et al., 2014) or DCNN-based autoencoder (Zhang et al., 2017).

5.3 Qualitative Analysis

5.3.1 Analysis of Semantic Information

To understand what information has been learned in our NASH model, we examine the matrix $E \in \mathbb{R}^{d \times l}$ in (6). Similar to (Miao et al., 2016; Larochelle and Lauly, 2012), we select the 5 nearest words according to the word vectors learned from NASH and compare with the corresponding results from NVDM.

As shown in Table 2, although our NASH model contains a binary latent variable, rather than a continuous one as in NVDM, it also effectively group semantically-similar words together in the learned vector space. This further demonstrates that the proposed generative framework manages to bypass the binary/discrete constraint and is able to abstract useful semantic information from documents.

5.3.2 Case Study

In Table 5, we show some examples of the learned binary hashing codes on 20Newsgroups dataset.
dataset. We observe that for both 8-bit and 16-bit cases, NASH typically compresses documents with shared topics into very similar binary codes. On the contrary, the hashing codes for documents with different topics exhibit much larger Hamming distance. As a result, relevant documents can be efficiently retrieved by simply computing their Hamming distances.

6 Conclusions

This paper presents a first step towards end-to-end semantic hashing, where the binary/discrete constraints are carefully handled with an effective gradient estimator. A neural variational framework is introduced to train our model. Motivated by the connections between the proposed method and rate-distortion theory, we inject data-dependent noise into the Bernoulli latent variable at the training stage. The effectiveness of our framework is demonstrated with extensive experiments.

Acknowledgments We would like to thank the ACL reviewers for their insightful suggestions. This research was supported in part by DARPA, DOE, NIH, NSF and ONR.

References

Johannes Ballé, Valero Laparra, and Eero P Simoncelli. 2016. End-to-end optimization of nonlinear transform codes for perceptual quality. In Picture Coding Symposium (PCS), 2016. IEEE, pages 1–5.

Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432.

Ting Chen, Martin Renqiang Min, and Yizhou Sun. 2017. Learning k-way d-dimensional discrete code for compact embedding representations. arXiv preprint arXiv:1711.03067.

Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. 2017. Stochastic generative hashing. arXiv preprint arXiv:1701.02815.

Mayur Datar, Nicole Immorlica, Piotr Indyk, and Virginia Mirrokni. 2004. Locality-sensitive hashing scheme based on p-stable distributions. In Proceedings of the twentieth annual symposium on Computational geometry. ACM, pages 253–262.

Geoffrey Hinton. 2012. Neural networks for machine learning. coursera. URL: http://coursera.org/course/neuralnets.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2016. Binarized neural networks. In Advances in neural information processing systems. pages 4107–4115.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Diederik P Kingma and Max Welling. 2013. Autoencoding variational bayes. arXiv preprint arXiv:1312.6114.

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. pages 3294–3302.

Yehuda Koren. 2008. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining. ACM, pages 426–434.

Hugo Larochelle and Stanislas Lauly. 2012. A neural autoregressive topic model. In Advances in Neural Information Processing Systems. pages 2708–2716.

Michael S Lew, Nicu Sebe, Chabane Djeraba, and Ramesh Jain. 2006. Content-based multimedia information retrieval: State of the art and challenges. ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM) 2(1):1–19.

Qi Li, Zhenan Sun, Ran He, and Tieniu Tan. 2017. Deep supervised discrete hashing. arXiv preprint arXiv:1705.10999.

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, pages 2074–2081.
Chris J Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712.

Christopher D Manning, Prabhakar Raghavan, Hinrich Schütze, et al. 2008. Introduction to information retrieval, volume 1. Cambridge university press Cambridge.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. arXiv preprint arXiv:1706.00359.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In International Conference on Machine Learning. pages 1727–1736.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositional- ity. In Advances in neural information processing systems. pages 3111–3119.

Sandeep Pandey, Andrei Broder, Flavio Chierichetti, Vanja Josifovski, Ravi Kumar, and Sergei Vassilvitskii. 2009. Nearest-neighbor caching for content-match applications. In Proceedings of the 18th international conference on World wide web. ACM, pages 441–450.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. International Journal of Approximate Reasoning 50(7):969–978.

Dinghan Shen, Renqi Min, Yitong Li, and Lawrence Carin. 2017a. Adaptive convolutional filter generation for natural language understanding. arXiv preprint arXiv:1709.08294.

Dinghan Shen, Guoyin Wang, Wenlin Wang, Martin Renqiang Min, Qinliang Su, Yizhe Zhang, Chunyuan Li, Ricardo Henao, and Lawrence Carin. 2018. Baseline needs more love: On simple word-embedding-based models and associated pooling mechanisms. In ACL.

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, and Lawrence Carin. 2017b. Deconvolutional latent-variable model for text sequence matching. AAAI.

Raphael Shu and Hideki Nakayama. 2017. Compressing word embeddings via deep compositional code learning. arXiv preprint arXiv:1711.01068.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research 15(1):1929–1958.

Benno Stein, Sven Meyer zu Eissen, and Martin Potthast. 2007. Strategies for retrieving plagiarized documents. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, pages 825–826.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Advances in neural information processing systems. pages 3104–3112.

Lucas Theis, Wenzhe Shi, Andrew Cunningham, and Ferenc Huszár. 2017. Lossy image compression with compressive autoencoders. ICLR.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In Advances in Neural Information Processing Systems. pages 6309–6318.

Jingdong Wang, Heng Tao Shen, Jingkuan Song, and Jianqiu Ji. 2014. Hashing for similarity search: A survey. arXiv preprint arXiv:1408.2927.

Jun Wang, Sanjiv Kumar, and Shih-Fu Chang. 2012. Semi-supervised hashing for large-scale search. IEEE Transactions on Pattern Analysis and Machine Intelligence 34(12):2393–2406.

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, pages 213–222.

Wenlin Wang, Zhe Gan, Wenqi Wang, Dinghan Shen, Jiaji Huang, Wei Ping, Sanjeev Satheesh, and Lawrence Carin. 2018. Topic compositional neural language model. In AISTATS.

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

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Convolutional neural networks for text hashing. In IJCAI. pages 1369–1375.

Zichao Yang, Zhitong Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. arXiv preprint arXiv:1702.08139.

Deli Zhang, Jun Wang, Deng Cai, and Jinsong Lu. 2010. Self-taught hashing for fast similarity search. In Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval. ACM, pages 18–25.

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. pages 4172–4182.