Doc2hash: Learning Discrete Latent Variables for Document Retrieval

Yifei Zhang
Vector Lab
JD Finance
zhangyifei11@jd.com

Hao Zhu
Vector Lab
JD Finance
zhuhao6@jd.com

Abstract

Learning to hash via generative models has become a powerful paradigm for fast similarity search in documents retrieval. To get binary representation (i.e., hash codes), the discrete distribution prior (i.e., Bernoulli Distribution) is applied to train the variational autoencoder (VAE). However, the discrete stochastic layer is usually incompatible with the backpropagation in the training stage and thus causes a gradient flow problem. In this paper, we propose a method, Doc2hash, that solves the gradient flow problem of the discrete stochastic layer by using continuous relaxation on priors, and trains the generative model in an end-to-end manner to generate hash codes. In qualitative and quantitative experiments, we show the proposed model outperforms other state of the art methods.

1 Introduction

A popular theme for deep learning is that of representation learning, whose goal is to use existing data to learn a compact and meaningful representation when building classifiers or other predictors. Learning continuous representation has been achieving a great success in various NLP tasks, including text classification, word and sentence representation (Mikolov et al., 2013; Le and Mikolov, 2014), yet learning discrete representation is potentially more suitable for tasks we are interested in such as learning to hash.

Hashing serves as a fast solution for similarity search also called approximate nearest neighbor search. In document retrieval, semantic hashing is the strategy that turn the document into binary codes (hash codes) which capture semantic information. One can use a query (a document) to retrieve similar documents by calculating the hamming distances between their hash codes. Given the fast calculation process and the efficient storage property (one modern PC can execute millions of hamming distance computations in just a few milliseconds). This semantic hashing strategy is very attractive.

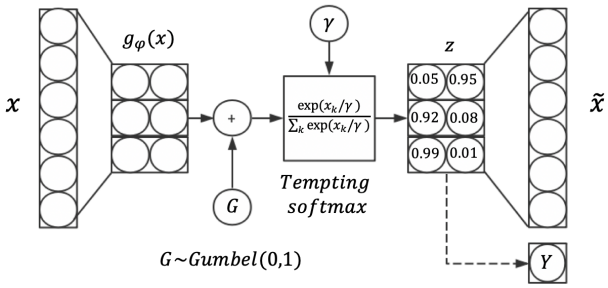


Figure 1: The end-to-end semantic hashing framework: Training the cycle $x \rightarrow z \rightarrow x$ to obtain the latent variables z. The one hot representation of discrete latent variables z has been relaxed to $\{(0.05, 0.95), (0.92, 0.08), (0.99, 0.01)\}$ by the tempting softmax. Hash codes $z_h = \{1, 0, 0\}$ is obtained via taking the $\arg\max$ of z. Y is the label/tag used in the supervised hashing

Inspired by the recent success of modeling latent variables via generative models in solving various NLP problems (van den Oord et al., 2017; Semeniuta et al., 2017; Bowman et al., 2015), Some approaches obtain binary codes of documents from a generative perspective via parameterizing models using neural networks and stochastic optimization using gradient-based techniques: (Chaidaroon and Fang, 2017) designed a two-stage training procedures to generate hash codes with variational autoencoder: (i) it first infers continuous representations of text through VAE with isotropic Gaussian distribution prior (ii) Obtain hash codes via binarizing the continuous representation of texts. Since the model parameters are not learned in an end-to-end manner, the two-stage training strategy may result in a suboptimal local optima. (Shen et al., 2018) replaced the Gaussian distribution prior with Bernoulli distribution prior so that the stochastic layer of the

Proceedings of NAACL-HLT 2019, pages 2235–2240
Minneapolis, Minnesota, June 2 - June 7, 2019.

VAE can directly produce binary codes in the latent space and train hash codes in an end-to-end manner. Unfortunately, the Bernoulli stochastic layer is non-differentiable. Although the Straight-Through (ST) estimator is adapted to propagate gradients where it skips the gradient of the stochastic layer during backpropagation (Bengio et al., 2013). The ST estimator is still a biased estimator which introduces high-variance biased gradients of objectives during the training.

In this paper, we propose a generative model with a categorical distribution prior for semantic hashing which learns binary codes of documents in an end-to-end manner. To train the generative model, instead of using the ST estimator, we use the Gumbel-Softmax trick to overcome the inability by applying the re-parameterization trick to discrete latent variables. There are two advantages: 1) a nice parameterization for a discrete (or categorical) distribution is given in terms of the Gumbel distribution (the Gumbel trick); and 2) although the corresponding function is non-continuous, it can be made continuous by applying using a continuous approximation that depends on a temperature parameter. Therefore, it produces low-variance biased gradients of the stochastic layer in the backpropagation. Our experiment shows our model achieves the state of the art performance on three standard datasets for semantic hashing.

2 Methodology

2.1 Discrete NVI Framework For Semantic Hashing

In this section, we introduce the neural variational inference (NVI) framework (Shen et al., 2018) with discrete latent variables for semantic hashing in detail. First, a generative model whose encoding distribution and decoding distribution are defined as $p(z|x)$, $p(x|z)$. We approximate these two distributions by defining approximations $q_\phi(z|x)$, $p_\theta(x|z)$ and model them as the inference (encoding) network $g_\phi(x)$ and the generative network (decoding) $g_\theta(z)$. $x \in R^V$ is the bag of word representation of a document where V is vocabulary size. Specially, the weight schema of bag of words can be binary, TF, TFIDF. The inference network $g_\phi(x)$ produces latent variables z from x approximating the true posterior distribution $p(z|x)$ as $q_\phi(z|x)$. Then, the generative network $g_\theta(z)$ maps latent variable z back to x by approximating $p(z|x)$ as $p_\theta(x|z)$. We model the conditional probability over the word w_i as softmax:

$$p(x_i = w_i|z) = \frac{\exp(w_i g_\theta(z))}{\sum_V \exp(w_i g_\theta(z))} \quad (1)$$

where $w_i \in \{0,1\}^V$ is represented as an one-hot vector indicating the i-th word in the total vocabulary and the likelihood $p_\theta(x|z) = \prod_i p(x_i = w_i|z)$. Training the generative model that synthesizes the observed data x, we can obtain the meaningful latent variables z.

To learn discrete latent variables under the NVI framework for semantic hashing, we cast z to be the discrete random variables and assume a multivariate categorical prior on z: $p(z) \sim \mathbf{Cat}(\pi) = \prod_i^l I(z = k)\pi_{ik}$, where π_{ik} is the k-th class probability on i-th component of parameters π. In this way, the posterior distribution approximated by the encoding network is constrained in the form of $q_\phi(z|x) = \mathbf{Cat}(h)$, where $h = g_\phi(x)$ is the output of the encoding network.

2.2 Variational Low Bound of Discrete Latent Variable Training

To train the parameters θ, ϕ of the encoding network $q_\theta(z|x)$ and the decoding network $p_\phi(x|z)$, we maximize the variational low bound as same as in the VAE framework (Kingma and Welling, 2013):

$$L_{ELBO} = E_{z\sim q_\phi(z|x)}[\log \frac{q_\theta(x|z)p(z)}{q_\phi(z|x)}] \quad (2)$$

$$= E_{z\sim q_\phi(z|x)}[\log q_\theta(x|z)] - KL(q_\phi(z|x)||p(z)) \quad (3)$$

the first term is the reconstruction loss, which encourages the decoder network to learn the mapping from latent variables z to the original document x. The second term is Kullback-Leibler (KL) divergence $KL(q_\phi(z|x)||p(z))]$, which encourages the posterior distribution $q_\phi(z|x)$ to be close to the multivariate categorical prior $p(z)$. we can write the KL term in the following form:

$$KL(q_\phi(z|x)||p(z)) = \sum_i \sum_k g_\phi^{ik}(x) \log \frac{g_\phi^{ik}(x)}{\pi_{ik}} \quad (4)$$

2.3 Gradients of Discrete Stochastic Layer

To evaluate $E_{z\sim q_\phi(z|x)}$, the reparameterization trick is often employed to make the sampling pro-

cess $z \sim q_\phi(z|x)$ compatible with backpropagation (Kingma and Welling, 2013). But sampling from a discrete distribution using reparameterization tricks still has problem during backpropagation. Due to the sampling from the stochastic layer usually incorporate hard threshold operators such as **sign(.)** and **arg max(.)** (Jang et al., 2016; Maddison et al., 2016), the stochastic layer becomes non-differentiable. In general, the solution to this issue is using the Straight-Through (ST) estimator where a biased path derivative estimator can be utilized even when z is not reparameterizable as shown in Figure 2. For example, NASH (Shen et al., 2018) passes the gradients through the Bernoulli stochastic layer using ST estimator. Specifically, they do:

$$\frac{dL}{d\phi} = \frac{dL}{dz}\frac{dz}{dg_\phi(x)}\frac{dg_\phi(x)}{d\phi} \approx \frac{dL}{dz}\frac{dg_\phi(x)}{d\phi} \quad (5)$$

where the ST estimator lets the gradients of the stochastic layer $\frac{dz}{dg_\phi(x)} \approx 1$. However, the ST estimator is backpropagating with respect to the sample-independent. It may cause discrepancies between the forward and backward pass, and thus lead to higher variance (Jang et al., 2016).

2.4 Continuous Relaxation of Discrete Stochastic Layer

As mention above, we want to learn discrete Latent variables (hash codes) by maximizing the variational low bound assuming a multivariate categorical prior. Supposed the output of the encoding network, $h = g_\phi(x)$, $h \in R^{l\times 2}$, represents the parameters of the multivariate categorical posterior $q(z|x)$. We can use reparameterization trick to draw samples of z from the categorical posterior using Gumbel distribution (Gumbel, 1954; Jang et al., 2016):

$$z_i = \arg\max_k(G_k + logh_i^k), k \in \{0, 1\} \quad (6)$$

G_1, G_2 is drawn i.i.d. from Gumbel (0, 1) = $\log(-\log(U(0, 1))$ where U is the uniform distribution. We further represent z_i as one-hot vector with 2 dimensions:

$$z_i = \text{OneHot}[\arg\max_k(G_k + logh_i^k)], k \in \{0, 1\} \quad (7)$$

discrete latent variables z can be seen as concatenation of l one-hot vectors.

In the forward pass, we sample z from the stochastic discrete layer via Eq. 7. However, in backpropagation, we cannot pass gradient through the hard threshold operator, $\arg\max$, since it is non-differentiable. The derivative of $\arg\max$ is 0 everywhere except the boundary of state change, where it is undefined. To end this, instead of employing ST estimator to skip gradients, we use the tempting softmax as a continuous relaxation of the $\arg\max$ computation:

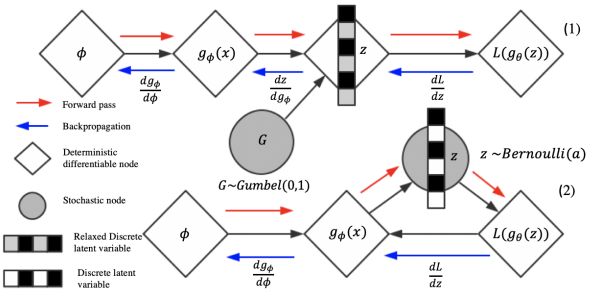


Figure 2: Gradients estimation in stochastic computation graphs (1) Gumbel-Softmax trick (2) The Straight-Through estimator, used for Bernoulli discrete variables, skips stochastic node by approximating $\frac{dz}{dg_\phi} \approx 1$

$$z_i = \frac{\exp((G_k + \log h_i^k)/\gamma)}{\sum_k \exp((G_k + \log h_i^k)/\gamma)}, k \in \{0, 1\} \quad (8)$$

This relaxation approximates the exactly discrete $\arg\max$ computation as temperature parameter $\gamma \to 0$ yet keeps the relative order of $(G_k + \log h_i^k)$. similar relaxation techniques have been introduced as the Gumbel-Softmax trick in (Jang et al., 2016; Maddison et al., 2016). Since Eq. 8 is differentiable, we can train the generative model in an end-to-end way via stochastic gradient descent to get the discrete latent variables (hash codes). The whole framework is summarized in the Figure 1 and the difference against the ST estimator is exemplified in the Figure 2.

2.5 Supervised Hashing

We extend our method to the supervised setting, where the mapping from latent variables z to the label y is learned, here parameterized by a two-layer multilayer perceptron (MLP) followed by a fully-connected softmax layer. To balance maximizing the variational lower bound and minimizing the discriminative loss, the following joint training objective is employed:

$$L = -L_{ELBO}(\theta, \phi, \boldsymbol{x}) + \alpha L_{dis}(\eta, \boldsymbol{z}, y) \quad (9)$$

where η refers to parameters of the MLP classifier and α controls the relative weight between the

variational lower bound (L_{ELBO}) and the discriminative loss (L_{dis}). We assume a general multi-label classification setting where each document could have multiple labels/tags. $P(y_j|f(\boldsymbol{z};\eta))$ can be modeled by the logistic function defined as:

$$P(y_j|f(\boldsymbol{z},\eta)) = \frac{1}{1+\exp(-y_j^T \eta_j \boldsymbol{z})} \quad (10)$$

3 Experiment

3.1 Baseline and Setting

	Unsupervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
LSH	0.4388	0.4393	0.4514	0.4553	0.4773
S-RBM	0.4846	0.5108	0.5166	0.5190	0.5137
SpH	0.5807	0.6055	0.6281	0.6143	0.5891
STH	0.3723	0.3947	0.4105	0.4181	0.4123
VDSH	0.4330	0.6853	0.7108	0.4410	0.5847
NASH-DN	0.6358	0.6956	0.7327	0.7010	0.6325
Doc2hash	**0.6965**	**0.7224**	**0.7473**	**0.7532**	**0.7595**
	Supervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
KSH	0.6608	0.6842	0.7047	0.7175	0.7243
SHTTM	0.6299	0.6571	0.6485	0.6893	0.6474
VDSH-SP	0.7498	0.7798	0.7891	0.7888	0.7970
NASH-DN-S	0.7438	0.7946	0.7987	0.801	0.8139
Doc2hash-S	**0.8140**	**0.8472**	**0.8490**	**0.8492**	**0.8439**

Table 1: Precision of the top 100 retrieved documents on TMC dataset.

	Unsupervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
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-DN	0.7470	0.8013	0.8418	0.8297	0.7924
Doc2hash	**0.7543**	**0.8102**	**0.8487**	**0.8361**	**0.8344**
	Supervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
KSH	0.7840	0.8376	0.8480	0.8537	0.8620
SHTTM	0.7992	0.8520	0.8323	0.8271	.8150
VDSH-SP	0.8890	0.9326	0.9283	0.9286	0.9395
NASH-DN-S	**0.9214**	0.9327	0.9455	0.9589	0.9502
Doc2hash-S	0.9134	**0.9338**	**0.9557**	**0.9602**	**0.9598**

Table 2: Precision of the top 100 retrieved documents on Reuters dataset.

We evaluate the proposed method named Doc2hash in both supervised and unsupervised setting for semantic hashing. For unsupervised task, these baselines are selected: Locality Sensitive Hashing (LSH) (Datar et al., 2004), Stack Restricted Boltzmann Machines (Salakhutdinov and Hinton, 2009), Spectral Hashing (Weiss et al., 2009), Self taught Hashing (STH) (Zhang et al., 2010), Variational Deep Semantic Hashing (VDSH) (Chaidaroon and Fang, 2017), and the best variant of Neural Architecture for Generative Semantic Hashing (Shen et al., 2018) (NASH-DN). For the supervised task, we also compare the proposed method with kinds of baselines: Supervised Hashing with Kernels (KSH) (Liu et al., 2012), Semantic Hashing using Tags and Topic Modeling (SHTTM) (Wang et al., 2013), Supervised VDSH (VDSH-SP) and Supervised NASH (NASH-DN-S).

	Unsupervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
LSH	0.4180	0.4352	0.4716	0.5214	0.5877
S-RBM	0.5106	0.5743	0.6130	0.6463	0.6531
SpH	0.5093	0.7121	0.7475	0.7559	0.7423
STH	0.3975	0.4898	0.5592	0.5945	0.5946
VDSH	0.7976	0.7944	0.8481	0.8951	0.8444
Doc2hash-S	**0.8495**	**0.8858**	**0.9001**	**0.9123**	**0.9167**
	Supervised Hashing				
Model	8bits	16bits	32bits	64bits	128bits
KSH	0.9126	0.9146	0.9221	0.9333	0.9350
SHTTM	0.8820	0.9038	0.9258	0.9459	0.9447
VDSH-SP	0.9666	0.9757	0.9788	**0.9805**	0.9794
Doc2hash-S	**0.9720**	**0.98001**	**0.9810**	0.9802	**0.9797**

Table 3: Precision of the top 100 retrieved documents on RCV1 dataset.

We follow the same experimental protocol and setting used in (Shen et al., 2018; Chaidaroon and Fang, 2017). Three standard public datasets of documents are chosen for training and evaluation: Reuters (Lewis, 1997), TMC (Oza, 2010), and RCV1 (Lewis et al., 2004). We evaluate the proposed model by calculating the precision of top 100 retrieved documents based on hamming distance of their hash codes. The final precision score is then averaged over all test documents. To make comparable results with previous works, we experiment with latent variables size of 8, 32, 64 and 128.

3.2 Semantic Hashing Evaluation

In this section, we evaluate Doc2hash over various number of bits on the three datasets. In the unsupervised hashing, Doc2hash outperforms NASH-DN, the current state of the art model for semantic hashing, and other methods. The Table 1 shows the result of TMC dataset. Doc2hash shows its ability to assign the similar data (with the same label) to the hash codes of which hamming distances are small. The same trend and superiority of Doc2hash are also observed in both Reuters and RCV1 datasets as shown in Tables 2 and 3. Note that we do not compare with NASH-DN in RCV1 because NASH doesn't report any results

on RCV1 dataset and doesn't release any code to reproduce their result. We compare the reminding methods with Doc2hash. Table 3 shows our approach improves the precision with significant margin compared with other models. We note that the retrieval results tend to drop when we set the length of hash codes to be 64 or larger, which also happens for some baselines. It is probably because of over-fitting. Compared with other methods, the proposed method is robust when the codes length increase. Our approach performs better than other methods in 64 and 128 bits setting, suggesting that Doc2hash can effectively generate hash codes to documents even with limited training data.

We also evaluate Doc2hash in the supervised hashing setting with the same datasets. We make use of the label/tag information during training. As shown in Tables 1, 2 and 3, Doc2hash yields better results than the other baseline models in different bits length.

3.3 Quantitative Analysis

In this section, we explore the reason why the proposed method outperforms the counterpart based on the ST estimator. We try to analyze the distribution of latent variables that trained with ST estimator (NASH) and Gumbel softmax relaxation (our method) respectively to quantitatively estimate the coding efficiency. To obtain efficient hash codes in the unsupervised setting, an expected pattern is that we get the same amount of 1 as well as 0. Because according to information theory, such a uniform distribution produces the maximum entropy in Bernoulli distribution. Motivated by this idea, we make a comparison between NASH and Doc2hash in the conditional probabilistic distribution of latent variables (i.e., $p(z|x)$). As we can see in Figure 4(a), the distribution of $p(z|x)$ in NASH is significantly asymmetric while the proposed method generates a well-balanced distribution (as Shown in Figure 4(b)). This comparison demonstrates that our approach is more close to the expected target in the probabilistic distribution of $p(z|x)$. Furthermore, we calculate the proportion of 1 in both methods. The proportion of NASH is 40%, while our method is 49%. Thus, it shows our approach is able to produce more efficient hash codes by using Gumbel softmax relaxation because it is very close to the up-bound of information theory for Bernoulli distribution. We demonstrate this point more clearly by showing the distribution of $g_\phi(x)$ in NASH. It is asymmetric because the peak value tilts to the left side. And quantization based on thresholding around the zero point implies that the hash codes produced by NASH contain more number of 0 than the number of 1.

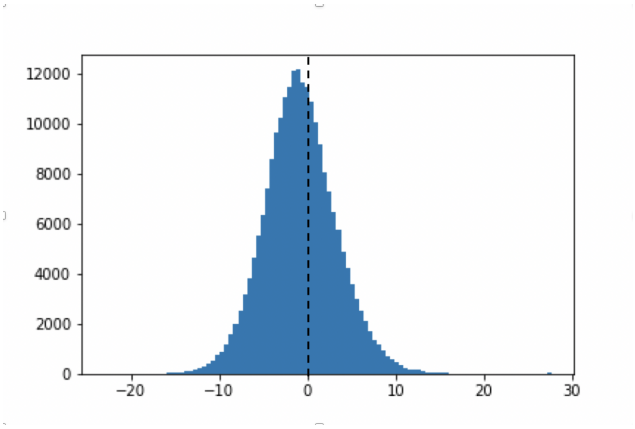
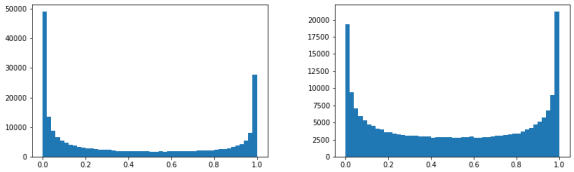


Figure 3: The distribution of $g_\phi(x)$ in NASH



(a) (b)

Figure 4: the distribution of $p(z|x)$: (a) is the distribution of NASH. (b) is the distribution of Doc2hash

4 Conclusion

In this paper, we present a generative model with discrete latent variables to learn binary representation for semantic hashing. We show that compared with the Straight-Through estimator, the Gumbel-Softmax relaxation provides a better solution to learn hash codes and thus gains the state of the art performance in three different datasets. We also explore the reason why the proposed method produces more efficient hash codes compared with the counterpart based on the ST estimator.

Acknowledgements

We sincerely thank anonymous reviewers for your constructive suggestions to improve the quality of this paper.

References

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

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

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*, pages 75–84. ACM.

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

Emil Julius Gumbel. 1954. Statistical theory of extreme values and some practical applications. *NBS Applied Mathematics Series*, 33.

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

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International Conference on Machine Learning*, pages 1188–1196.

David D. Lewis. 1997. Reuters-21578 text categorization collection data set.

David D Lewis, Yiming Yang, Tony G Rose, and Fan Li. 2004. Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr):361–397.

Wei Liu, Jun Wang, Rongrong Ji, Yu-Gang Jiang, and Shih-Fu Chang. 2012. Supervised hashing with kernels. In *CVPR*, pages 2074–2081. IEEE.

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*.

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

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Nikunj Oza. 2010. Siam 2007 text mining competition dataset.

Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. *IJAR*, 50(7):969–978.

Stanislau Semeniuta, Aliaksei Severyn, and Erhardt Barth. 2017. A hybrid convolutional variational autoencoder for text generation. *arXiv preprint arXiv:1702.02390*.

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)*, pages 2041–2050. Association for Computational Linguistics.

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*, pages 213–222. ACM.

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

Dell 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*, pages 18–25. ACM.