Rank Based Bernoulli Variational Autoencoder for Semantic Hashing

Aravind Dendukuri[1][0000−0002−7826−7114], Jannat Arora[2][0000−0002−1208−3603], Abhinav Pradeep[1][0000−0001−8367−4702], and Karanth Shyam Subraya[1][0000−0003−3232−4213]

[1] Manipal Institute of Technology, Manipal, India
[2] Thapar Institute of Engineering and Technology
{denarvi,jannatarora21,abhinavp2301}@gmail.com, shyam.karanth@manipal.edu

Abstract. The past decade has fostered a massive burst in the amount of actionable text-based data. This rapid increase calls for efficient search and retrieval techniques in large-scale information systems. Semantic hashing is a novel proposition to inexpensively tackle high volume tasks. It was developed to perform context-based document retrieval and employs a powerful strategy that converts data documents to compact binary hash codes. Variational Autoencoders (VAEs) have been a critical component of the generative models often used in semantic hashing. The discrete nature of the binary latent variables generated by the encoder allows it to captures the context of the data sample better than a continuous distribution. However, training discrete variables proves to be inefficient as standard backpropagation through the variables is infeasible. We propose a solution to this problem through the continuous relaxation of the latent variables using Gumbel-Softmax and training the model in an end-to-end fashion. The hashes generated, however, are representative of only the individual document. Hence, a ranking component is incorporated to take care of the inter-document dependency. Weak supervision is used to train this ranking component leveraging the volume of cheap unlabeled data.
Experimentation on four publicly available datasets shows that our proposed model outperforms the current unsupervised state of the art models.

Keywords: Gumbel-softmax · semantic hashing · generative models · ranking · information retrieval

1 Introduction

Nearest neighbour search also called as Semantic Search, has varied applications in the field of large-scale information retrieval where simple linear search across the large-scale corpora would be computationally infeasible. Hence, a class of compact and computationally efficient methods are required instead. Semantic hashing [4] is a technique of converting the semantics of the source document

to binary code consisting only of 1s and 0s. It finds its usage in Plagiarism Detection [1], Content-based Document Finding [2], Near duplicate detection [3].

Similarity search [5] is performed on these hashed binary codes using the Hamming distance measure, which is the number of differing bits between the binary codes of the documents being compared. Similar documents will have similar binary codes and subsequently lower deltas. Given the speed of the modern PC to perform millions of bit-based calculations in a fraction of a second, Semantic hashing proves itself a fast, memory light storage efficient solution.

Historically, hashing algorithms have been segmented as either data-dependent or data-independent. Data-dependent techniques usually create codes of much longer length than the data-independent models and rely on the random projections of the input probability functions of the source corpora. Locality Sensitive Hashing (LSH) [6] and its many variants like Kenelized Locality Sensitive hashing (KLSH) [7], super-bit hashing and non-metric LSH [8] are examples of the data-dependent hashing techniques. At its simplest, LSH works by hashing data points into buckets such that points near each other are also located in the same buckets with a high probability [9]. The simplicity of the technique lends it to high processing speeds but it also has many disadvantages; The training data is not used at all, learning efficiency is low, and as established before, longer hash codes need to be generated to perform the similarity search. Due to these limitations, data-independent hashing techniques are primarily used for providing fast initial approximations.

However, these limitations have given rise to methods leveraging data-centric modern machine learning techniques. At the cost of training time, machine learning and deep learning techniques can optimize the embedding space by creating deep generative models for hashing. [10].

Modern deep learning-based approaches tackle hashing by obtaining the binary representation of the documents from a generative perspective using stochastic gradient-based optimisations. Variational Autoencoder (VAE) [11] is one such generative model that has found remarkable success in this domain since it provides efficient methods for probabilistic inference and can be scaled for massive data sets. The Variational Deep Semantic Hashing model (VDSH) [12], an established baseline for this task, is created through a VAE as well. Its code generation is done in two parts: 1) Use of a Gaussian prior to infer continuous latent variables through a VAE 2) Binarization of the continuous latent variables. Since this method was not trained in an end-to-end manner, it may result in a sub-optimal solution. Further, since conventional VAEs are taught using a Gaussian prior, the continuous variables generated by the encoder need to be quantized, leading to quantization errors creeping in that could hinder the performance of the information retrieval process. Subsequent models [13] replace the Gaussian prior with a Bernoulli distribution prior to generate binary codes as latent variables that could be trained in an end-to-end fashion. However, they encounter difficulties in training their models since backpropagation is not possible through discrete variables. Straight through estimator [14] has been proposed

as one possible method to get around this differentiation problem. However, the ST estimator has been shown to introduce high variance during the training.

Lastly, contemporary models are tasked to generate document dependent hashes without much attention on effective similarity search. To this end, we propose an unsupervised generative VAE model that can be trained end-to-end and can directly incorporate document similarities through the attachment of a ranking module on top of our VAE network. End-to-end trainability can be achieved by replacing the ST estimator with Gumbel-Softmax [15,16] that allows us to backpropagate through discrete variables.

The ranking component uses weak supervision [17] to obtain pseudo rankings, and triplet loss is applied to enforce that similar documents will have similar hash codes. Both components – the VAE and the ranking module, are trained together. Experimentation across four publicly available datasets show that our model can attain the state of the art performance across different hash code lengths.

The rest of the paper is organized as follows. We discuss related work in the following section. In Section 3, we discuss our model and our methodology. In Section 4, we present the results of our model on different datasets [18–20] and finally, section 5 contains the conclusion and scope for future work.

2 RELATED WORK

2.1 Semantic Hashing

Hashing can be defined as the process of mapping a higher dimensional query to a lower-dimensional value to enable ease in searching. In the case of Semantic hashing, the hashing algorithm creates an embedding function h(x) that maps the feature space X of the input corpora to a Hamming Space $H = \{0, 1\}^m$, with m being the number of bits in the resultant hash. Searches are hence, performed in the hashing space instead of the input features as they allow for highly optimal nearest neighbour lookups using the Hamming distance. Incorporating the effectiveness of binary codes, hashing methods have been widely used for similarity searching [5] - the process of searching a document set D to find the elements closest to some query document q. Research has shown that performing similarity search using the binary hashes produces results orders of magnitudes faster compared to conventional search techniques even via basic search algorithms.

In comparison to the hashing algorithms like LSH [6], modern Semantic hashing algorithms adopt the data-dependent hashing objective using Machine learning algorithms. Spectral hashing [21], constraining the balanced bits with the uncorrelated bits, can achieve the document similarity preservation objective. Graph hashing [22] exploits the intrinsic manifold structure of data captured by a graph representation to achieve the similarity objective. Self-taught hashing [23] creates the hashes using a two-pronged approach: 1) Find the optimal l-bit binary codes for all documents in the given corpus via Laplacian Eigenmap in an unsupervised setting. 2) Train l classifiers via supervised learning to predict the l-bit code for any unseen query document.

Salakhutdinov et al. [10] seminal work seems to be the first reported use of deep generative models for hashing. Their use of stacked Boltzmann machines modelled the performance of a stochastic Auto encoder where thresholding the latent variable nodes generated the hash codes. Given the difficulty in training such a model, future research opted for simpler Deep learning models.

Recent generative models incorporate Variational Autoencoder [12] for the hashing challenge and threshold the continuous latent variables around the data median by computing the discrete variables using an indicator function. However, discretization by thresholding brings with it quantization error that is calculated as $||h(x) - \phi(x)||$, where h(x) is the discrete output after thresholding, and $\phi(x)$ is the continuous output embedding. This error has a significant impact on the search performance. This model, though improved upon the performance of previous works and proves to be more stable in training than Salakhutdinov. Following this, NASH [13] proposed a fully differential end-to-end model that learns the binarized bits directly without the second step needed to discretize the generated vectors. This is achieved by treating the binary hashing codes as Bernoulli latent variables. This technique forms the basis for most subsequent work [24–26]. Though, since the binarization component is non-differentiable, a straight- through component is used.

2.2 Unsupervised ranking methods

Based on the works of STH (maintaining document connectivity but excluding document features) and VDSH (preserving document content but not the neighbourhood), Chaidaroon et al. [17] proposed an autoencoder model similar to VDSH that uses the BM25 ranking model as its weak annotator. The weak signals from the BM25 model [27] are used to estimate the true document space and reconstruct the average neighbourhood documents, thereby preserving local semantic similarities. In contrast, Ranking Based Semantic Hashing (RBSH) [25] proposes a hinge loss-based weakly supervised model, that uses a small amount of labelled data, large quantities of unlabeled data, and a ranking component to optimize both local and global structures. RBSH achieves this by taking document triplets (d, d1, d2) as inputs with pre-calculated similarities using STH hash and then using weak supervision to predict either d1 or d2 as being the most similar to d.

3 MODEL ARCHITECTURE AND METHODOLOGY

We use the Bernoulli VAE, proposed by [24] as our base model to learn the hash codes $\{0, 1\}^m$. A standard VAE has, as its latent variables, a continuous Gaussian representation $\mathcal{N}(\mu(x), \sigma(x))$ where $\mu(x)$ and $\sigma(x)$ are modelled by a neural network $f(\mathrm{x}; \phi(x))$. In our model, the latent variables are instead binary and assumed to be Bernoulli distributed.

Binary latent variables are preferable to continuous variables for two reasons: Interpretability and removal of the quantization-based error. The relationship

set up between the hash codes and the representation learnt is unambiguous compared to a continuous Gaussian representation, and with the hash codes generated being binary by default, no thresholding operation is required. This removes the error inducing step from the model pipeline leading to better search performance. However, with discrete variables arises the problem of optimization. This problem is solved by re-parameterization using Gumbel-SoftMax. We shall now describe the components of our model pipeline.

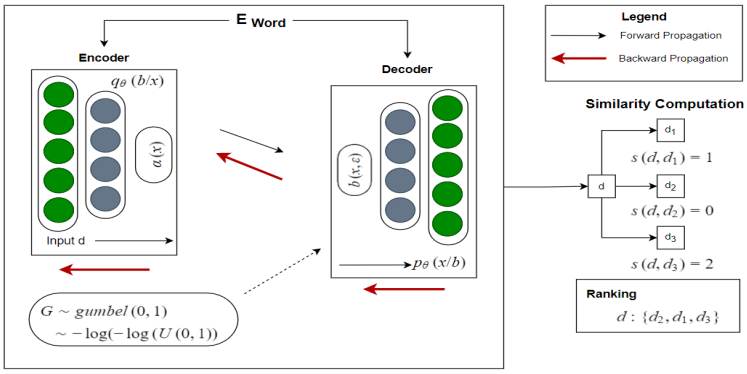

Fig. 1: Model Architecture

3.1 Variational framework

We have based our model on the architecture proposed by RBSH [25] and BinaryVAE [24]. A bag of words representation, of vocabulary size V, is assumed for the document d in the input corpora such that each unique word is represented as W_d. A multi-variate Bernoulli distribution $B(\sigma(d))$, through the encoder network $q_\phi(b|d)$ is considered for the binary latent variable, b. The probability $\sigma(d) = p(b = 1|d)$ is conducted using the network $f(\mathrm{d}; \phi)$. A decoder function, $p_\theta(d|b)$ is then tasked to reconstruct the input from the latent variable b.

The reconstruction objective is defined as a multinomial distribution on the input tokens and learnt through a neural network.

$$p(d|b) = \prod_{w \in d} p(w|b)^{n_w}$$

n_w is the word frequency

3.2 Variational Loss

The encoder, $q_\phi(b|d)$ and the decoder $p_\theta(d|b)$ are jointly trained together as part of a stochastic Autoencoder on the parameters ϕ and θ respectively by maximizing the data log-likelihood. However, training proves to be intractable since

the input, d is unobserved. Hence, VAEs in general are trained by maximizing the variational lower bound of $l(\phi, \theta)$

$$l(\phi, \theta) = E_{(q_\phi(b|d))}[logp_\theta(d|b)] - D_{kl}(q_\phi(b|d)||p_\theta(b)$$

The former term is the reconstruction loss, encouraging the decoder function to map the latent binary variable to the input d. The latter term is the Kullback-Leibler (KL) divergence which enforces the relation between the posterior associated with our encoder function $q_\phi(b|d)$ and our chosen prior $p_\theta(b)$.

Normal evaluation of these terms is done through the reparameterization method. However, that proves to be insufficient in case of discrete variables such as in our case. This is due to the presence of hard threshold functions - sign(.) and arg max(.) in the sampling step that makes the stochastic layer non-differentiable. One possible solution to this problem is the use of straight through estimators. The ST estimator ignores the derivative of the threshold function and directly passes on the incoming gradient as if it passed through an identity function.

This, however, leads to an output with high-variance due to discrepancies between the backward and forward pass [15]. Hence, a more concentrated solutions must be implemented.

The Gumbel-softmax distribution [15, 16] allows for a continuous relaxation of the discrete Bernoulli Latent variables.

In the forward pass

$$b_i = OneHot[argmax_k(G_k + log(h_i^k))], k \in 0, 1$$

h is output of the encoder function. $G_i \sim \mathcal{U}(0, 1)$

Since, the gradient can't be passed backward through the hard argmax(.) function, during the back-propagation part we can approximate the latent variables as:

$$\hat{b_{i,l}} = \sigma((log\frac{\alpha_i(x^{(l)})}{1 - \alpha_i(x^{(l)})} + log\frac{G_i}{1 - G_i})/\lambda)$$

With $b_{i,l} \sim Ber(\alpha_i(x^l)), \sigma(x) = 1/(1 + exp(-x))$ and $\forall i \in [B]$

λ is the temperature parameter and can be tuned. With $\lambda \to 0$ equating the discrete Argmax computation. Through this $\hat{b_{i,l}}$ can be made to approximate the real Bernoulli samples $b_{i,l}$. According to [15,16] a good value for the temperature parameters λ is 2/3.

3.3 Prior

Based on the evaluation by Binary VAE [24], we choose the Bernoulli distribution $p_\theta(b_i) = Ber(0.5) \forall i \in [B]$ as our prior. This assumes balanced hash codes: half active and half inactive. With this the second term of our loss function, the KL divergence, can be calculated as:

$$D_{kl} = \sum_{i}^{B} E_{(q_\phi(b_i|d))}[logq_\phi(b_i|d)] - E_{(q_\phi(b_i|d))}[logp_\theta(b_i)]$$

$$= B.log2 + \sum_{i}^{B} \alpha_i(d).log\alpha_i(d) + (1 - \alpha_i(d)).log(1 - \alpha_i(d))$$

We will now describe our encoder, decoder, and ranking components as pictured in fig. 1.

3.4 Encoder and Decoder

The encoder $q_\phi(b|d)$ transforms the input document to its hash codes by sampling the output through a Bernoulli distribution. To compute the latent codes, a representation is generated using the importance embedding.

$$v1 = ReLU(Wa(d \odot Eimp) + ba)$$

$\odot$ corresponds to elementwise multiplication, W and b are weight matrices and bias vectors, respectively. The importance embedding scales the word level values of the original document such that words of lesser value have a lower impact on the final hash code.

We need to sample the encoder to obtain the hash codes since it is stochastic in nature. RBSH obtains the final codes by estimating a vector $\mu = [\mu_1, \mu_2, ..., \mu_m]$ of values sampled uniformly at random from the interval [0, 1] and computing each bit value of either 0 or 1 as:

$$b_i = [Q(d)_i - \mu_i]$$

In our method since we chose our prior $b \sim Ber(\alpha(x))$, binary codes are assured with no required discretization. However, in case deterministic codes are required a threshold of 0.5 can be used, with $b = 1(\alpha(x) - 0.5)$.

The decoder model is based on [25] with the same importance embedding as our encoder but with a symmetric model that was proposed in [10] to improve our performance.

3.5 Ranking Component

To ensure that similarity is maintained between two similar documents we introduce a ranking component to the model. To this end, the triplet loss is calculated. For input triplets, (d, d1, d2) we find the pairwise similarities of d1 and d2 with respect to d. The hash codes generated by Self- Taught hashing are chosen for this purpose. Document similarity is calculated as the Euclidean distance between the two hashes. The top k similar documents are thus found using k-nearest neighbour algorithm.

A modified hinge loss function, as proposed by RBSH [25], is chosen to compute this ranking loss.

3.6 Combination of VAE network and ranking component

The variational and ranking components are trained simultaneously by minimizing their combined loss function.

$$L = \alpha L_{rank} - \beta l(\phi, \theta)$$

α and β are used to scale the ranking and variational components, respectively.

Method	**8 Bits**	**16 Bits**	**32 Bits**	**64 Bits**	**8 Bits**	**16 Bits**	**32 Bits**	**64 Bits**
STH	0.6544	0.7245	0.7505	0.7684	0.6744	0.7218	0.7432	0.7512
VDSH	0.6576	0.6987	0.7463	0.7495	0.6330	0.6853	0.7108	0.6510
NASH	0.6202	0.7068	0.7644	0.7798	0.6802	0.7368	0.7644	0.7998
RBSH	0.7021	0.7251	0.7784	0.7864	0.7492	0.7621	0.7940	0.8163
Our Method	0.7366	0.7782	0.8266	0.8140	0.7776	0.7982	0.8290	0.8188
	Reuters				**TMC**			
Method	**8 Bits**	**16 Bits**	**32 Bits**	**64 Bits**	**8 Bits**	**16 Bits**	**32 Bits**	**64 Bits**
STH	0.4044	0.4743	0.5572	0.5753	0.6255	0.7717	0.8163	0.8239
VDSH	0.7230	0.7403	0.7948	0.7980	0.6298	0.6715	0.6906	0.7144
NASH	0.7642	0.7718	0.8148	0.8319	0.7242	0.7798	0.8048	0.8185
RBSH	0.7592	0.7974	0.8384	0.8476	0.7953	0.8144	0.8330	0.8369
Our Method	0.7610	0.8184	0.8589	0.8520	0.8097	0.8097	0.8470	0.8452
	RCV1				**AGNews**			

Table 1: The precision of the top 100 retrieved documents on the Rueters, TMC, RCV1 and AGNews datasets respectively across different hash lengths.

4 EXPERIMENTAL EVALUATION

We evaluate our model on four widely used datasets on text retrieval task: Reuters21578, containing 11,000 news documents; TMC2, dataset of NASA air traffic reports; RCV1 containing more than 800,000 manually categorized newswire reports and AGNews, dataset of a more than a million news articles containing articles for world, sports, business and science/tech.

4.1 Preprocessing and Training

The most frequent words are removed from the document as part of the preprocessing. Additionally, non- text characters, and stop words are also removed. The text is then converted to lower-case and converted to a TF-IDF representation.

A 75, 15, 10 split is performed on the input to attain the training, test, and validation set, respectively. After training on the training set, documents from the test or the validation set are passed as queries and a Knn (K-nearest neighbor) search is performed on the hash codes generated. This is based on the principle of fast semantic search using Hamming distance where the number of differing bits in the generated hash dictates the closeness of two documents. For a query document, K is taken as 100 for a Knn search and the average precision for all the test document is marked.

4.2 Baseline

We compare the performance of our model against the current state of the art models described earlier: 1) Self-taught hashing [23], 2) VDSH [12], 3) NASH [13], 4) RBSH [25]. These models are trained as per the parameter tuning instructions provided in their original paper.

4.3 Result

We present the result of our experimentation in the above tables. Hash code of length {8, 16, 32, 64} are compared based on the Prec@100 metric and the best performance is highlighted.

4.4 Result Evaluation

In this section, we evaluate the results of our model shown in the table. Our model is shown to outperform the baselines in almost all scenarios. We analyze the distribution of latent variables that have been trained with the ST estimator via NASH versus the Gumbel Softmax in the case of our model to quantify the coding efficiency.

According to information theory, a pattern of equivalent 1's and 0's would prove to be the most efficient hash code as such a uniform distribution would produce the maximum entropy in a Bernoulli distribution. Additionally, by adopting a ranking based semantic hashing methodology, our approach outperforms both traditional machine learning based solutions (STH, LCH) and other recent neural models (VDSH, NASH).

To further evaluate the impact of the ranking based hashing, the variance was calculated both with and without the ranking component. Both models were evaluated over a variety of datasets like Reuters and AGnews and internal tests consistently tracked better performance in the model with the ranking component. Interestingly, the deltas varied in prominence across different bit lengths with a significant jump from 8 to 16 bits. However, as the bit size grew, the differences proportionately diminished until marginal. This further helps highlight the effectiveness of the ranking component while dealing with smaller hashes.

To further quantify the delta between the hash codes generated by modern neural approaches like NASH and the ones generated by our proposed method,

the proportions of 1's in both the hashes are compared. The quantization around the zero suggests that the NASH hash codes contain a higher 0 count. During our tests, we've observed that hashes generated by the traditional solutions tend to have erratic proportions. This in turn leads to a lack of consistent performance across different use cases and datasets. The neural solutions, in turn, have a more consistent track record across the various benchmarks. However our proposed method, as showcased, achieves similar if not better accuracy with hash codes a factor of 2-4x bit smaller then the traditional baseline models. This grants a compounding increment in performance on large scale similarity searches.

Finally, almost counter-intuitively, use of higher number of bits does not equate better results. The performance, in-fact, seems to drop as the number of bits increases. This could be because of over-fitting. In general, the best result seems to be achieved when the number of bits is set to 32.

5 Conclusion

In this paper, we have presented a novel deep learning based generative model for semantic hashing incorporating the variational component through the use of a Bernoulli VAE that can directly map the input to a binary hash code without the need of the thresholding. Thus, eliminating the quantization error. We have further implemented a ranking mechanism to learn to rank the documents based on their similarity on the principle that similar documents would have similar hash codes as well. The ranking component allows us to incorporate both local and global document structure in the generated hash codes.

We show that the use of the Gumbel softmax function eliminates the variance caused by a straight-through estimator and allows for complete back-propagation of the loss value. Thus, our model can be trained in a completely end-to-end manner.

Experimentation on the four publicly available datasets show that our model is able to achieve better performance than both the baselines and the previous state of the art semantic hashing models.

Future work would involve investing the effects of using different encoder-decoder architectures, that have been inspired by image-hashing models, for our use case. Further, performance of our model on large scale Image retrieval tasks will be tested.

References

1. Ceglarek D., Haniewicz K. (2012) Fast Plagiarism Detection by Sentence Hashing. In: Rutkowski L., Korytkowski M., Scherer R., Tadeusiewicz R., Zadeh L.A., Zurada J.M. (eds) Artificial Intelligence and Soft Computing. ICAISC 2012. Lecture Notes in Computer Science, vol 7268. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29350-4_4
2. M. S. Lew, N. Sebe, C. Djeraba, and R. Jain. Content-based multimedia information retrieval: State of the art and challenges. ACM Transactions on Multimedia Computing

3. Gurmeet Singh Manku, Arvind Jain, and Anish Das Sarma. Detecting near-duplicates for web crawling. In WWW, pages 141–150. ACM, 2007.
4. Ruslan Salakhutdinov and Geoffrey Hinton. 2009. Semantic hashing. International Journal of Approximate Reasoning 50, 7 (2009), 969–978
5. Christoph Mangold,A survey and classification of semantic search approaches,Int. J. Metadata Semant. Ontologies, 2. pages 23-34, 2007
6. Alexandr Andoni and Piotr Indyk. Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. In FOCS, pages 459–468, 2006
7. B. Kulis and K. Grauman. Kernelized localitysensitive hashing for scalable image search. In Computer Vision, 2009 IEEE 12th International Conference on, pages 2130–2137. IEEE, 2009
8. Ji, Jianqiu and Li, Jianmin and Yan, Shuicheng and Zhang, Bo and Tian, Qi. Super-Bit Locality-Sensitive Hashing. Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 1, pages 108-116, 2012
9. https://santhoshhari.github.io/Locality-Sensitive-Hashing/
10. Salakhutdinov, R., Hinton, G.: Semantic hashing. Int. J. Approximate Reasoning 50(7), 969–978 (2009)
11. Kingma, D.P., Welling, M.: Auto-encoding variational Bayes (2013)
12. Suthee Chaidaroon and Yi Fang. 2017. Variational deep semantic hashing for text documents. In ACM SIGIR Conference on Research and Development in Information Retrieval. pages 75–84.
13. Dinghan Shen and Qinliang Su and Paidamoyo Chapfuwa and Wenlin Wang and Guoyin Wang and L. Carin and Ricardo Henao, NASH: Toward End-to-End Neural Architecture for Generative Semantic Hashing, ArXiv 2018
14. Yoshua Bengio, Nicholas Leonard, and Aaron ´ Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. arXiv preprint arXiv:1308.3432. 2013
15. Jang, E., Gu, S., Poole, B.: Categorical reparameterization with Gumbel-softmax. In: Proceedings of the ICLR (2017)
16. Maddison, C.J., Mnih, A., Teh, Y.W.: The concrete distribution: a continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712 (2016)
17. Chaidaroon, Suthee and Ebesu, Travis and Fang, Yi. Deep Semantic Text Hashing with Weak Supervision. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. 2018. Pages 1109 1112
18. Reuters dataset https://archive.ics.uci.edu/ml/datasets/reuters-21578+text+categorization+collection
19. TMC dataset https://catalog.data.gov/dataset/siam-2007-text-mining-competition-dataset
20. RCV1 dataset https://scikit-learn.org/0.18/datasets/rcv1.html
21. Weiss, Yair and Torralba, Antonio and Fergus, Rob. Spectral Hashing. Advances in Neural Information Processing Systems 2009
22. Liu, Wei and Mu, Cun and Kumar, Sanjiv and Chang, Shih-Fu. Discrete Graph Hashing. Advances in Neural Information Processing Systems. 2014
23. Zhang, Dell and Wang, Jun and Cai, Deng and Lu, Jinsong. Self-Taught Hashing for Fast Similarity Search. Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2010. Pages 18 25
24. Mena F., Ñanculef R. (2019) A Binary Variational Autoencoder for Hashing. In: Nyström I., Hernández Heredia Y., Milián Núñez V. (eds) Progress in Pattern Recognition, Image Analysis, Computer Vision, and Applications.

CIARP 2019. Lecture Notes in Computer Science, vol 11896. Springer, Cham. https://doi.org/10.1007/978-3-030-33904-3_12
25. Hansen, Casper and Hansen, Christian and Simonsen, Jakob Grue and Alstrup, Stephen and Lioma, Christina. Unsupervised Neural Generative Semantic Hashing. Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval. 2019. Pages 735–744
26. Zhang, Yifei and Zhu, Hao. Doc2hash: Learning Discrete Latent variables for Documents Retrieval. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers). 2019. Pages 2235 - 2240
27. Amati G. (2009) BM25. In: LIU L., ÖZSU M.T. (eds) Encyclopedia of Database Systems. Springer, Boston, MA. https://doi.org/10.1007/978-0-387-39940-9_921