Efficient Document Retrieval by End-to-End Refining and Quantizing BERT Embedding with Contrastive Product Quantization

Zexuan Qiu[1], **Qinliang Su**[1,2]*, **Jianxing Yu**[3], **and Shijing Si**[4]
[1]School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou, China
[2] Guangdong Key Laboratory of Big Data Analysis and Processing, Guangzhou, China
[3]School of Artificial Intelligence, Sun Yat-sen University, Guangzhou, China
[4]School of Economics and Finance, Shanghai International Studies University, China
{qiuzx3@mail2, suqliang@mail, yujx26@mail}.sysu.edu.cn

Abstract

Efficient document retrieval heavily relies on the technique of semantic hashing, which learns a binary code for every document and employs Hamming distance to evaluate document distances. However, existing semantic hashing methods are mostly established on outdated TFIDF features, which obviously do not contain lots of important semantic information about documents. Furthermore, the Hamming distance can only be equal to one of several integer values, significantly limiting its representational ability for document distances. To address these issues, in this paper, we propose to leverage BERT embeddings to perform efficient retrieval based on the product quantization technique, which will assign for every document a real-valued codeword from the codebook, instead of a binary code as in semantic hashing. Specifically, we first transform the original BERT embeddings via a learnable mapping and feed the transformed embedding into a probabilistic product quantization module to output the assigned codeword. The refining and quantizing modules can be optimized in an end-to-end manner by minimizing the probabilistic contrastive loss. A mutual information maximization based method is further proposed to improve the representativeness of codewords, so that documents can be quantized more accurately. Extensive experiments conducted on three benchmarks demonstrate that our proposed method significantly outperforms current state-of-the-art baselines[1].

1 Introduction

In the era of big data, Approximate Nearest Neighbor (ANN) search has attracted tremendous attention thanks to its high search efficiency and extraordinary performance in modern information retrieval systems. By quantizing each document as a compact binary code, semantic hashing (Salakhutdinov and Hinton, 2009) has become the main solution to ANN search due to the extremely low cost of calculating Hamming distance between binary codes. One of the main approaches for unsupervised semantic hashing methods is established on generative models (Chaidaroon and Fang, 2017; Shen et al., 2018; Dong et al., 2019; Zheng et al., 2020), which encourage the binary codes to be able to reconstruct the input documents. Alternatively, some other methods are driven by graphs (Weiss et al., 2008; Chaidaroon et al., 2020; Hansen et al., 2020; Ou et al., 2021a), hoping the binary codes can recover the neighborhood relationship. Though these methods have obtained great retrieval performance, there still exist two main problems.

Firstly, these methods are mostly established on top of the outdated TFIDF features, which do not contain various kinds of important information of documents, like word order, contextual information, etc. In recent years, pre-trained language models like BERT have achieved tremendous success in various downstream tasks. Thus, a natural question to ask is whether we can establish efficient retrieval methods on BERT embeddings. However, it has been widely reported that BERT embeddings are not suitable for semantic similarity-related tasks (Reimers and Gurevych, 2019), which perform even worse than the traditional Glove embeddings (Pennington et al., 2014). (Ethayarajh, 2019; Li et al., 2020) attribute this to the "anisotropy" phenomenon that BERT embeddings only occupy a narrow cone in the vector space, causing the semantic information hidden in BERT embeddings not easy to be leveraged directly. Thus, it is important to investigate how to effectively leverage the BERT embeddings for efficient document retrieval.

Secondly, to guarantee the efficiency of retrieval, most existing methods quantize every document to a binary code via semantic hashing. There is

*Corresponding author.
[1]Our PyTorch code is available at https://github.com/qiuzx2/MICPQ, and our MindSpore code will be also released soon.

Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 853 - 863
December 7-11, 2022

no doubt that the Hamming distance can improve the retrieval efficiency significantly, but it also restricts the representation of document similarities seriously since it can only be an integer from $-B$ to B for B-bit codes. Recently, an alternative approach named product quantization (Jégou et al., 2011; Jang and Cho, 2021; Wang et al., 2021) has been proposed in the computer vision community to alleviate this problem. Basically, it seeks to quantize every item to one of the codewords in a codebook, which is represented by a Cartesian product of multiple small codebooks. It has been shown that product quantization is able to deliver superior performance than semantic hashing while keeping the cost of computation and storage relatively unchanged. However, this technique has rarely been explored in unsupervised document retrieval.

Motivated by the two problems above, in this paper, we propose an end-to-end contrastive product quantization model to jointly refine the original BERT embeddings and quantize the refined embeddings into codewords. Specifically, we first transform the original BERT embeddings via a learnable mapping and feed the transformed embedding into a probabilistic product quantization module to output a quantized representation (codeword). To preserve as much semantic information as possible in the quantized representations, inspired by recent successes of contrastive learning, a probabilistic contrastive loss is designed and trained in an end-to-end manner, simultaneously achieving the optimization of refining and quantizing modules. Later, to further improve the retrieval performance, inspired by the recent development of clustering, a mutual information maximization based method is further developed to increase the representativeness of learned codewords. By doing so, the cluster structure hidden in the dataset of documents could be kept soundly, making the documents be quantized more accurately. Extensive experiments are conducted on three real-world datasets, and the experimental results demonstrate that our proposed method significantly outperforms current state-of-the-art baselines. Empirical analyses also demonstrate the effectiveness of every proposed component in our proposed model.

2 Preliminaries of Product Quantization for Information Retrieval

In fields of efficient information retrieval, a prevalent approach is semantic hashing, which maps every item x to a binary code b and then uses Hamming distances to reflect the semantic similarity of items. Thanks to the low cost of computing Hamming distance, the retrieval can be performed very efficiently. However, the Hamming distance can only be an integer from $-B$ to B for B-bit codes, which is too restrictive to reflect the rich similarity information.

An alternative approach is vector quantization (VQ) (Gray and Neuhoff, 1998), which assigns every item with a codeword from a codebook C. The codeword in VQ could be any vector in $\mathbb{R}^D$, rather than limited to the binary form as in semantic hashing. By storing pre-computed distances between any two codewords in a table, the distance between items can be obtained efficiently by looking up the table. However, to ensure competitive performance, the number of codewords in a codebook needs to be very large. For example, there are 2^{64} different codes for a 64-bit binary code, and thus the number of codewords in VQ should also be of this scale, which however is too large to be handled.

To tackle this issue, product quantization (Jégou et al., 2011) proposes to represent the codebook C as a Cartesian product of M small codebooks

$$C = C^1 \times C^2 \times \cdots \times C^M, \tag{1}$$

where the m-th codebook C^m consists of K codewords $\{c_k^m\}_{k=1}^K$ with $c_k^m \in \mathbb{R}^{D/M}$. For an item, the product quantization will choose a codeword from every codebook C^m, and the final codeword assigned to this item is

$$c = c^1 \circ c^2 \cdots \circ c^M, \tag{2}$$

where c^m denotes the codeword chosen from C^m, and $\circ$ denotes concatenation. For each codeword c, we only need to record its indices in the M codebooks, which only requires $M \log_2 K$ bits. Thanks to the Cartesian product decomposition, now we only need to store MK codewords of dimension $\mathbb{R}^{D/M}$ and M lookup tables of size $K \times K$. As an example, to enable a total of 2^{64} codewords, we can set $M = 32$ and $K = 4$, which obviously will reduce the size of footprint and lookup tables significantly. During retrieval, we only need to look up the M tables and sum them up, which is only slightly more costly than the computation of Hamming distance.

3 The End-to-End Joint Refining and Quantizing Framework

To retrieve semantic-similar documents, a core problem is how to produce for every document a quantized representation that preserves the semantic-similarity information of original documents. In this section, a simple two-stage method is first proposed, and then we develop an end-to-end method that is able to directly output representations with desired properties.

3.1 A Simple Two-Stage Approach

To obtain semantics-preserving quantized representations, a naive approach is to first promote the semantic information in original BERT embeddings and then quantize them. Many methods have been proposed on how to refine BERT embeddings to promote their semantic information. These methods can be essentially described as transforming the original embedding $z(x)$ into another one $\tilde{z}(x)$ via a mapping $g(\cdot)$ as

$$\tilde{z}(x) = g(z(x)), \tag{3}$$

where $g(\cdot)$ could be a flow-based mapping (Li et al., 2020), or a neural network trained to maximize the agreement between representations of a document's two views (Gao et al., 2021), etc. It has been reported that the refined embeddings $\tilde{z}(x)$ are semantically much richer than original one $z(x)$. Then, we can follow standard product quantization procedures to quantize the refined embeddings $\tilde{z}(x)$ into discrete representations.

3.2 End-to-End Refining and Quantizing via Contrastive Product Quantization

Obviously, the separation between the refining and quantizing steps in the two-stage method could result in a significant loss of semantic information in the final quantized representation. To address this issue, an end-to-end refining and quantizing method is proposed. We first slice the original BERT embedding $z(x)$ into M segments $z^m(x) \in \mathbb{R}^{D/M}$ for $m = 1, 2, \cdots, M$. Then, we refine $z^m(x)$ by transforming it into a semantic-richer form $\tilde{z}^m(x)$ via a mapping $g^m_\theta(\cdot)$, that is,

$$\tilde{z}^m(x) = g^m_\theta(z^m(x)), \tag{4}$$

where the subscript θ denotes the learnable parameter. Different from the mapping $g(\cdot)$ which is determined at the refining stage and is irrelevant to the quantization in the two-stage method, the mapping $g^m_\theta(\cdot)$ here will be learned later by taking the influences of quantization error into account. Now, instead of quantizing the refined embedding $\tilde{z}^m(x)$ to a fixed codeword, we propose to quantize it to one of the codewords $\{c^m_{k^m}\}_{k^m=1}^K$ by stochastically selecting k^m according to the distribution

$$p(k^m|x) = \frac{\exp\left(-\|\tilde{z}^m(x) - c^m_{k^m}\|^2\right)}{\sum_{i=1}^K \exp\left(-\|\tilde{z}^m(x) - c^m_i\|^2\right)} \tag{5}$$

with $k^m = 1, 2, \cdots, K$. Obviously, the probability that $\tilde{z}^m(x)$ is quantized to a codeword is inversely proportional to their distance. Thus, by denoting k^m as a random sample drawn from $p(k^m|x)$, *i.e.*, $k^m \sim p(k^m|x)$, we can represent the m-th quantized representation of document x as

$$h^m(x) = C^m \cdot one\_hot(k^m), \tag{6}$$

and the whole quantized representation of x as

$$h(x) = h^1(x) \circ h^2(x) \cdots \circ h^M(x). \tag{7}$$

Note that the quantized representation $h(x)$ depends on random variables $k^m \sim p(k^m|x)$ for $m = 1, 2, \cdots, M$, thus $h(x)$ itself is also random.

Now, we seek to preserve as much semantic information as possible in the quantized representation $h(x)$. Inspired by the recent successes of contrastive learning in semantic-rich representation learning (Gao et al., 2021), we propose to minimize the contrastive loss. Specifically, for every document x, we first obtain two BERT embeddings by passing it through BERT two times with two independent dropout masks and then use the embeddings to generate two quantized representations $h^{(1)}(x)$ and $h^{(2)}(x)$ according to (6) and (7). Then, we define the contrastive loss as

$$\mathcal{L}_{cl} = -\frac{1}{|\mathcal{B}|}\sum_{x\in\mathcal{B}}\left(\ell^{(1)}(x) + \ell^{(2)}(x)\right), \tag{8}$$

where $\mathcal{B}$ denotes a mini-batch of training documents; and $\ell^{(i)}(x)$ for $i = 1, 2$ is defined as

$$\ell^{(i)}(x) \triangleq \log \frac{\mathcal{S}(h^{(1)}_x, h^{(2)}_x)}{\mathcal{S}(h^{(1)}_x, h^{(2)}_x) + \sum\limits_{\substack{t\in\mathcal{B}\setminus x \\ n=1,2}} \mathcal{S}(h^{(i)}_x, h^{(n)}_t)}, \tag{9}$$

with $h^{(1)}_x$ denoting the abbreviation of $h^{(1)}(x)$ for conciseness; and $\mathcal{S}(h_1, h_2)$ is defined as

$$\mathcal{S}(h_1, h_2) \triangleq \exp\left(sim(h_1, h_2)/\tau_{cl}\right), \tag{10}$$

with $sim(h_1, h_2) \triangleq \frac{h_1^T h_2}{\|h_1\|\|h_2\|}$ being the cosine similarity function, and τ_{cl} being a temperature hyperparamter.

Under the proposed quantization method above, the quantized representation $h(x)$ depends on the random variable $k^m \sim p(k^m|x)$, making it not deterministic w.r.t. a given document x. Thus, we do not directly optimize the random contrastive loss $\mathcal{L}_{cl}$, but minimize its expectation

$$\overline{\mathcal{L}}_{cl} = -\frac{1}{|\mathcal{B}|} \sum_{x \in \mathcal{B}} \left(\overline{\ell}^{(1)}(x) + \overline{\ell}^{(2)}(x) \right), \quad (11)$$

where $\overline{\ell}^{(i)}(x)$ represents the expectation of $\ell^{(i)}(x)$ w.r.t. $k^m \sim p(k^m|x)$ for $m = 1, 2, \cdots, M$, that is,

$$\overline{\ell}^{(i)}(x) = \mathbb{E}_{k^1, k^2, \cdots, k^M} \left[\ell^{(i)}(x) \right]. \quad (12)$$

Obviously, it is impossible to derive an analytical expression for $\overline{\ell}^{(i)}(x)$, making the optimization of $\overline{\mathcal{L}}_{cl}$ not feasible. To address this issue, it has been proposed in (Jang et al., 2017) that the random sample k^m drawn from distribution $p(k|x) = \frac{\exp\left(-\left\|\tilde{z}^m(x) - c_k^m\right\|^2\right)}{\sum_{i=1}^K \exp\left(-\left\|\tilde{z}^m(x) - c_i^m\right\|^2\right)}$ can be re-parameterized as

$$k^m = \arg\max_i \left[-\left\|\tilde{z}^m(x) - c_i^m\right\|^2 + \xi_i \right], \quad (13)$$

where ξ_i denote i.i.d. random samples drawn from the Gumbel distribution $Gumbel(0, 1)$. Then, using the softmax function to approximate the $\arg\max(\cdot)$, the m-th quantized representation $h^m(x)$ can be approximately represented as

$$\widetilde{h}^m(x) = C^m \cdot v, \quad (14)$$

where $v \in \mathbb{R}^K$ is a probability vector whose k-th element is

$$v_k = \frac{\exp\left(-\frac{\left\|\tilde{z}^m(x) - c_k^m\right\|^2 + \xi_k}{\tau}\right)}{\sum_{i=1}^K \exp\left(-\frac{\left\|\tilde{z}^m(x) - c_i^m\right\|^2 + \xi_i}{\tau}\right)}, \quad (15)$$

with τ being a hyper-parameter. It can be easily seen that $\widetilde{h}^m(x)$ will converge to $h^m(x)$ as $\tau \to 0$, thus $\widetilde{h}^m(x)$ can be used as a good approximation to $h^m(x)$. With the approximation, $\overline{\ell}^{(i)}(x)$ in (12) can be approximately written as

$$\overline{\ell}^{(i)}(x) \approx \log \frac{\mathcal{S}(\widetilde{h}_x^{(1)}, \widetilde{h}_x^{(2)})}{\mathcal{S}(\widetilde{h}_x^{(1)}, \widetilde{h}_x^{(2)}) + \sum\limits_{\substack{t \in \mathcal{B} \setminus x \\ n=1,2}} \mathcal{S}(\widetilde{h}_x^{(i)}, \widetilde{h}_t^{(n)})}, \quad (16)$$

where $\widetilde{h}_x^{(1)}$ is the abbreviation of $h^{(1)}(x)$. Substituting (16) into (11) gives an approximate analytical expression of $\overline{L}_{cl}$ that is differentiable w.r.t. the parameter θ for refining and codebooks $\{C^m\}_{m=1}^M$ for quantization. Therefore, we can optimize the θ and codebooks $\{C^m\}_{m=1}^M$ in an end-to-end manner, explicitly encouraging the quantized representations $h(x)$ to preserve more semantic information.

It is worth noting that the injected Gumbel noise ξ_i in (15) is important to yield a sound approximation $\overline{\ell}^{(i)}(x)$ in (16). Theoretically, the approximated $\overline{\ell}^{(i)}(x)$ is guaranteed to converge to the exact value when $\tau \to 0$ and a large number of independent Gumbel nosies are used to approximate the expectation. However, if we abandon this noise in the computation of v_k, we will lose the appealing property above. Our experimental results also demonstrate the advantages of injecting Gumbel noises in the approximation. Another point worth pointing out is that if the refined embedding $\tilde{z}^m(x)$ is quantized to the closest codeword deterministically, that is, letting $k^m = \arg\max_i \left[-\left\|\tilde{z}^m(x) - c_i^m\right\|^2 \right]$, then it becomes equivalent to our probabilistic quantization approach without using Gumbel noise, further demonstrating the advantages of our method.

3.3 Improving the Representativeness of Codewords via MI Maximization

It can be seen that the codewords in C^m work similarly to the cluster centers in clustering. The clustering centers are known to be prone to get stuck at suboptimal points, which applies to the codewords analogously. If the codewords are not representative enough, they could result in a significant loss of semantic information in the quantized representations (Ge et al., 2013; Cao et al., 2016).

It has been recently observed that maximizing mutual information (MI) between the data and the cluster assigned to it can often lead to much better clustering performance (Hu et al., 2017; Ji et al., 2019; Do et al., 2021). Inspired by this, to increase the representativeness of codewords, we also propose to maximize the MI between the original document x and the codeword (index) assigned to it. To this end, given the conditional distribution $p(k^m|x)$ in (5), we first estimate the marginal distribution of codeword index k^m as

$$p(k^m) \approx \frac{1}{|\mathcal{D}|} \sum_{x \in \mathcal{D}} p(k^m|x), \quad (17)$$

where $\mathcal{D}$ denotes the training dataset. Then, by definition, the entropy of the codeword index k^m can be estimated as

$$H(K^m) = -\sum_{k^m=1}^{K} p(k^m) \log\left(p(k^m)\right). \quad (18)$$

Similarly, the conditional entropy of codeword k^m given data x can be estimated as

$$H(K^m|X) = -\frac{1}{|\mathcal{D}|} p(k^m|x) \log(p(k^m|x)). \quad (19)$$

Now, the mutual information between codeword index k^m and data x can be easily obtained by definition as $I(X, K^m) = H(K^m) - H(K^m|X)$. In practice, we find that it is better not to directly maximize the MI $I(X, K^m)$, but to maximize its variant form

$$I(X, K^m) = H(K^m) - \alpha H(K^m|X), \quad (20)$$

where α is a non-negative hyper-parameter controlling the trade-off between two entropy terms. Intuitively, maximizing the MI can be understood as encouraging only one codeword is assigned a high probability for a document x, while all codewords are used evenly overall. Given the mutual information, the final training objective becomes

$$\mathcal{L} = \overline{\mathcal{L}}_{cl} - \lambda \sum_{m=1}^{M} I(X, K^m), \quad (21)$$

where λ is a hyper-parameter controlling the relative importance of the MI term. Since this method employs MI to improve the quality of codewords, we name the model as **MICPQ**, i.e., **M**utual-**I**nformation-Improved **C**ontrastive **P**roduct **Q**uantization.

4 Related Work

As the main solution to efficient document retrieval, unsupervised semantic hashing has been studied for years. Many existing unsupervised hashing methods are established on the generative models encouraging the binary codes to reconstruct the original document. For example, VDSH (Chaidaroon and Fang, 2017) proposes a two-stage scheme, in which it first learns the continuous representations under the VAE (Kingma and Welling, 2014) framework, and then cast them into the binary codes. To tackle the two-stage training issue, NASH (Shen et al., 2018) presents an end-to-end generative hashing framework where the binary codes are treated as Bernoulli latent variables, and introduces the Straight-Through (Bengio et al., 2013) estimator to estimate the gradient w.r.t. the discrete variables. Dong et al. (2019) employs the mixture priors to empower the binary code with stronger expressiveness, therefore resulting in better performance. Further, CorrSH (Zheng et al., 2020) employs the distribution of the Boltzmann machine to introduce correlations among the bits of binary codes. Ye et al. (2020) proposes the auxiliary implicit topic vectors to address the issue of information loss in the few-bit scenario. Also, a handful of recent works focus on how to inject the neighborhood information of the graph under the VAE framework. Chaidaroon et al. (2018); Hansen et al. (2020) seeks to learn the optimal binary codes that can reconstruct neighbors of original documents. A ranking loss is introduced in (Hansen et al., 2019) to accurately characterize the correlation between documents. Ou et al. (2021a) first proposes to integrate the semantics and neighborhood information with a graph-driven generative model.

Beyond generative hashing methods, studies on hashing via the mutual information (MI) principle emerges recently. AMMI (Stratos and Wiseman, 2020) learns a high-quality binary code by maximizing the MI between documents and binary codes. DHIM (Ou et al., 2021b) first compresses the BERT embeddings into binary codes by maximizing the MI between global codes and local codes from documents.

Another efficient retrieval mechanism is product quantization. The Product quantization (PQ) (Jégou et al., 2011) and its improved variants such as Optimized PQ (Ge et al., 2013) and Locally Optimized PQ (Kalantidis and Avrithis, 2014) are proposed to retrain richer distance information than hashing methods while conducting the retrieval efficiently. These shallow unsupervised PQ methods are often based on the well-trained representation and learn the quantization module with heuristic algorithms (Xiao et al., 2021), which often can not achieve satisfactory performance. In this paper, we propose an unsupervised MI-improved end-to-end unsupervised product quantization model MICPQ. We notice that the proposed MICPQ is somewhat similar to recent works w.r.t PQ (Jang and Cho, 2021; Wang et al., 2021) in the computer vision community. However, (Jang and Cho, 2021) focuses on analyzing the performance difference between different forms of contrastive losses, while

we focus on how to design an end-to-end model to jointly refine and quantize the BERT embedding via contrastive product quantization from a probabilistic perspective. (Wang et al., 2021) concentrates on how to improve the codeword diversity to prevent model degeneration by regularizer design; whereas there is no "model degeneration" phenomenon observed in our model, and we use the mutual information maximization based method to increase the representativeness of codewords to further improve the retrieval performance.

5 Experiments

5.1 Datasets, Evaluation and Baselines

Datasets The proposed MICPQ model is evaluated on three benchmark datasets, including NYT (Tao et al., 2018), AGNews (Zhang et al., 2015) and DBpedia (Lehmann et al., 2015). Details of the three datasets can be found in Appendix A.

Evaluation Metrics For every query document from the testing dataset, we retrieve its top-100 most similar documents from the training set with the Asymmetric distance computation (Jégou et al., 2011) which is formulized in Appendix B. Then the retrieval precision is calculated as the ratio of the relevant documents. Note that a retrieved document is considered relevant to the query if they both share the same label. Finally, the retrieval precision averaged over all test documents is reported.

Baselines We consider the following unsupervised deep semantic hashing methods for comparision: VDSH (Chaidaroon and Fang, 2017), NASH (Shen et al., 2018), BMSH (Dong et al., 2019), CorrSH (Zheng et al., 2020), WISH (Ye et al., 2020), AMMI (Stratos and Wiseman, 2020), DHIM (Ou et al., 2021b). For the reported performances of baselines, they are quoted from DHIM (Ou et al., 2021b). Apart from existing baselines, we also implement the following two baselines for a more thorough comparison.

AEPQ. We are interested in optimizing our product quantizer with the reconstructing objective. Specifically, the quantized semantics-preserving representation vector $h(x)$ is expected to reconstruct the original input features (i.e., BERT embeddings) with a newly added decoder, which is similar with previous generative hashing methods. Same as MICPQ, the performance of AEPQ is evaluated with the Asymmetric distance computation. We name this baseline as AEPQ, i.e., **A**uto-**E**ncoder-based **P**roduct **Q**uantization.

CSH. Same as the encoder setting in NASH (Shen et al., 2018), we assume that the binary codes are generated by sampling from the multivariate Bernoulli distribution and propose to minimize the expected contrastive loss w.r.t. the discrete binary codes directly. The straight-through (ST) gradient estimator (Bengio et al., 2013) is utilized for training in an end-to-end manner. The data augmentation strategy is the same as that of our MICPQ. We name the proposed baseline as CSH, i.e., **C**ontrastive **S**emantic **H**ashing.

Training Details In our MICPQ, to output the refined vectors that will be fed into the product quantizer with the desired dimension, the encoder network is constituted by a pre-trained BERT Module followed by an one-layer ReLU feedforward neural network on top of the [CLS] representation whose dimension is 768. For performances under the average pooling setting of BERT, please refer to Appendix C. During the training, following the setting in DHIM (Ou et al., 2021b), we fix the parameters of BERT, while only training the newly added parameters. We implement the proposed model with PyTorch and employ the Adam optimizer(Kingma and Ba, 2015) for optimization.

In terms of hyper-parameters relevant to the product quantization, the dimension of codeword $\frac{D}{M}$ in the small codebook C^m is fixed to 24 and the number of codewords K in each codebook C^m is fixed to 16. By setting the number of small codebooks M as $\{4, 8, 16, 32\}$, we can see that the final codeword in the codebook C can be represented by $\{16, 32, 64, 128\}$ bits according to $B = Mlog_2K$. Thus, when compared with semantic hashing, they are compared under the same number of bits.

For other hyper-parameters, the learning rate is set as 0.001; the dropout rate p_{drop} to generate positive pairs for contrastive learning is set as 0.3; the Gumbel-Softmax temperature τ is set as 10 for 16-bit binary codes and 5 for longer codes; the temperature τ_{cl} in contrastive learning is set as 0.3; the trade-off coefficient α in (20) is set as 0.1; the coefficient λ in (21) is chosen from $\{0.1, 0.2, 0.3\}$ according to the performance on the validation set.

5.2 Results and Analyses

5.2.1 Overall Performance

Table 1 presents performances of our proposed model and existing baselines on three public

Table 1: The precision(%) comparision with different state-of-the-art unsupervised efficient retrieval methods. Among them, bold numbers represent best performance, and underlined numbers represent second best performance.

Method	NYT				AGNews				DBpedia			
	16bits	32bits	64bits	128bits	16bits	32bits	64bits	128bits	16bits	32bits	64bits	128bits
Using TFIDF features												
VDSH	68.77	68.77	75.01	78.49	67.32	67.42	72.70	73.86	67.79	72.64	78.84	84.91
NASH	74.87	75.52	75.08	73.01	65.74	69.34	72.72	74.33	78.02	79.84	79.79	76.76
WISH	70.15	70.03	64.48	68.94	74.53	74.79	75.05	72.70	82.82	82.76	82.10	78.22
BMSH	74.02	76.38	76.88	77.63	74.09	76.03	76.09	73.56	83.17	86.24	87.05	83.86
CorrSH	75.43	77.61	77.24	78.39	76.20	76.45	76.61	77.67	82.01	81.78	80.94	85.77
AMMI	71.06	76.48	77.37	78.03	76.47	76.61	77.32	78.23	84.51	89.53	90.78	91.03
Using BERT [CLS] embeddings												
VDSH	53.38	58.18	62.44	64.64	62.97	66.35	69.57	70.27	69.59	75.21	79.54	80.62
NASH	55.87	58.25	60.98	64.27	66.32	68.44	70.40	72.07	65.87	74.54	77.96	81.43
WISH	58.83	64.75	65.47	70.34	65.35	66.19	69.39	72.03	65.65	72.91	76.66	82.29
BMSH	59.35	63.26	65.87	69.71	66.77	69.61	71.99	73.16	66.42	79.13	82.01	84.57
CorrSH	62.03	65.48	68.38	72.28	67.06	68.51	70.86	73.17	65.28	74.63	78.65	83.61
AMMI	60.47	65.10	69.67	74.47	65.50	68.26	71.85	74.36	80.25	82.67	89.26	86.74
DHIM	79.69	80.55	79.77	79.09	78.23	79.17	78.88	79.86	94.26	94.80	93.02	88.21
AEPQ	67.98	68.35	70.38	71.48	68.81	70.49	72.82	73.84	80.24	84.54	86.78	89.70
CSH	79.63	81.05	81.08	81.28	79.14	80.25	81.14	81.78	95.19	95.84	95.79	96.01
MICPQ	**83.15**	**84.02**	**84.24**	**85.08**	**80.36**	**81.84**	**82.93**	**83.29**	**96.51**	**97.00**	**97.03**	**97.21**

datasets with code lengths varying from 16 to 128. It can be seen that the proposed simple baseline CSH achieves promising performances across all three datasets and nearly all code length settings when compared to previous state-of-the-art methods, demonstrating the superiority of using contrastive learning to promote semantic information. Further, our proposed MICPQ outperforms the previous methods by a more substantial margin. Specifically, improvements of 4.32%, 3.07% and 4.37% averaged over all code lengths are observed on NYT, AGNews and DBpedia datasets, respectively, when compared with the current state-of-the-art DHIM. Moreover, the performance of AEPQ lags behind our proposed MICPQ remarkably, which illustrates the limitation of the reconstructing objective. It is also observed that the retrieval performance of our proposed MICPQ consistently improves as the code length increases. Although this is consistent with our intuition that a longer code can preserve more semantic information, it does not always hold in some previous models (e.g., NASH, BMSH, DHIM).

5.2.2 Abalation Study

To understand the influence of different components in the MICPQ, we further evaluate the retrieval performance of two variants of MICPQ. *(i)* MICPQ_{cl}: it removes the mutual-information term $I(X, K^m)$ in each codebook and only optimizes the quantized contrastive loss to learn the semantics-preserving quantized representation; *(ii)* $\text{MICPQ}_{\text{softmax}}$: it does not inject the Gumbel noise, but only utilizes the sole softmax operation to produce the deterministic codeword index. As seen from Table 2, when compared to MICPQ_{cl}, our MICPQ improves the retrieval performance averaged over all code lengths by 1.51% and 0.94% on NYT and AGNews respectively, demonstrating the effectiveness of our mutual-information term inside each codebook. Also, by comparing MICPQ to $\text{MICPQ}_{\text{softmax}}$, consistent improvements can be observed on both datasets, which demonstrates the superiority of the proposed probabilistic product quantization.

Table 2: The precision (%) comparision with variants of MICPQ.

Ablation Study		16bits	32bits	64bits	128bits
NYT	MICPQ_{cl}	81.77	82.28	82.96	83.63
	$\text{MICPQ}_{\text{softmax}}$	82.28	83.46	83.64	84.17
	MICPQ	**83.15**	**84.02**	**84.24**	**85.08**
AGNews	MICPQ_{cl}	79.68	80.97	81.79	82.23
	$\text{MICPQ}_{\text{softmax}}$	79.60	81.61	82.58	82.69
	MICPQ	**80.36**	**81.84**	**82.93**	**83.29**

5.3 Link to Semantic Hashing: A Special Form of MICPQ

Through an extreme setting, the MICPQ model can be reconsidered in the semantic hashing framework and be evaluated using the Hamming distance rather than the Asymmetric distance. Specifically,

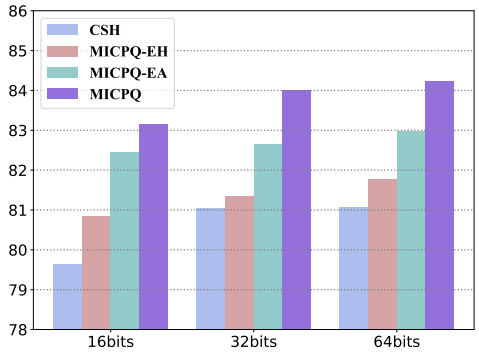

Figure 1: The precision(%) on NYT w.r.t. MICPQ under the extreme setting.

Table 3: Average and maximal accuracy(%) over M codebooks on NYT.

	Avg Acc			Max Acc		
	NYT	AGNews	DBpedia	NYT	AGNews	DBpedia
KMeans	41.73	**61.10**	**67.99**	43.74	67.45	80.49
MICPQ	**48.74**	58.06	64.12	**57.51**	**69.65**	**80.88**

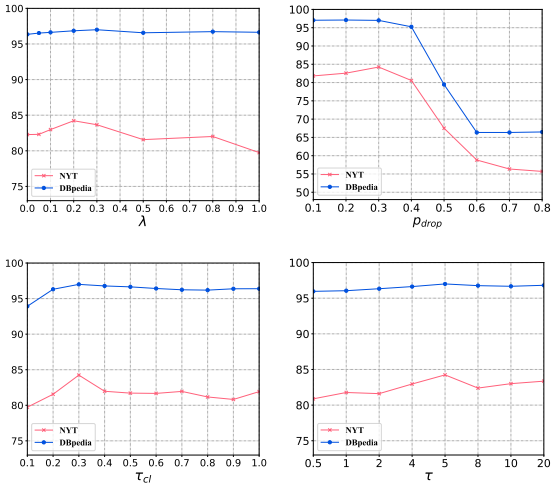

Figure 2: Hyper-parameter analyses with the precision(%) of 32-bits setting on NYT and DBpedia.

we push the number of codebooks M equal to its maximal limit (i.e., the code length B), and the number of codewords K inside each codebook is forced as 2 to satisfy the equation $B = M \log_2 K$. This way, the state of each bit in the B-bit binary code will be decided by a sub-space with 2 codewords. Under this extreme setting, we can either evaluate MICPQ with the Hamming distance evaluation or the Asymmetric distance. We name both models as MICPQ-EH (i.e., **E**xtreme-**H**amming) and MICPQ-EA (i.e., **E**xtreme-**A**symetric). As shown in Figure 1, the MICPQ-EA consistently outperforms the MICPQ-EH thanks to the superiority of Asymmetric Distance computation. Also, MICPQ-EH can be seen as the semantic hashing model since it's evaluated with Hamming distance, and the better retrieval performance of MICPQ-EH when compared to CSH informs us that we can take the special form of MICPQ as one of the excellent semantic hashing methods. To sum up, the proposed MICPQ is more flexible and powerful than the existing hashing baselines in terms of efficient document retrieval.

5.4 Evaluating the Quality of Codewords from the Clustering Perspective

To examine how well the codewords can represent the corresponding refined vectors $\tilde{z}^m(x)$, we set the number of codewords K in each codebook as the number of the ground-truth classes on datasets (i.e., $K = 26, 14$ and 4 on NYT, DBpedia and AGNews respectively), so that we can compute the unsupervised clustering accuracy with the help of the Hungarian algorithm (Kuhn, 1955) in each codebook. The number of codebooks M is set as 8 on all datasets. For comparison, we also run K-Means on each $\tilde{z}^m(x)$ separately. As shown in Table 3, the codewords in our MICPQ are on par with the ones learned by K-Means. Particularly, our MICPQ significantly outperforms K-means on NYT that has the largest number of classes (i.e., 26), demonstrating the ability of our MICPQ to learn high-quality codewords, especially for datasets with diverse categories.

5.5 Sensitive Analyses of Hyper-Parameters

We investigate the influence of 4 key hyper-parameters: the coefficient λ, the dropout rate p_{drop}, the temperature τ_{cl} in contrastive learning, and the Gumbel-Softmax temperature τ. As shown in Figure 2, compared with the case of $\lambda = 0$, obvious performance gains can be observed by introducing the mutual-information loss inside each codebook when the λ is set as a relatively small value (e.g., 0.2 or 0.3). The value of p_{drop} controls the strength of data augmentation. We see that as p_{drop} exceeds some thresholds (e.g., 0.4), the performance will decrease sharply, and eventually the model will collapse on both datasets. Also, it is shown that as τ_{cl} grows up, the precision first increases and reaches the peak when τ_{cl} is around 0.3 on both datasets. As for the Gumbel-Softmax temperature τ, we suggest setting it to a larger value, saying [5,15].

6 Conclusion

In this paper, we proposed a novel unsupervised product quantization model, namely MICPQ. In MICPQ, we managed to develop an end-to-end probabilistic contrastive product quantization model to jointly refine the original BERT embeddings and quantize the refined embeddings into codewords, with a probabilistic contrastive loss designed to preserve the semantic information in the quantized representations. Moreover, to improve the representativeness of codewords for keeping the cluster structure of documents soundly, we proposed to maximize the mutual information between data and the codeword assignment. Extensive experiments showed that our model significantly outperformed existing unsupervised hashing methods.

7 Limitations

In our work, we do not analyze the individuality of codebooks and the difference between codebooks. However, each codebook should show different aspects of characteristics for preserving rich semantics. Therefore, it may help if the difference between codebooks is more analyzed and improved this way. Also, in the future we may try to apply this method to the tasks of passage retrieval, on which some large-scale datasets for evaluation are available.

Acknowledgement

This work is supported by the National Natural Science Foundation of China (No. 62276280, U1811264), Key R&D Program of Guangdong Province (No. 2018B010107005), National Natural Science Foundation of Guangdong Province (No. 2021A1515012299), Science and Technology Program of Guangzhou (No. 202102021205). This work is also supported by CAAI-Huawei MindSpore Open Fund.

References

Yoshua Bengio, Nicholas Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

Yue Cao, Mingsheng Long, Jianmin Wang, Han Zhu, and Qingfu Wen. 2016. Deep quantization network for efficient image retrieval. In *AAAI*, pages 3457–3463. AAAI Press.

Suthee Chaidaroon, Travis Ebesu, and Yi Fang. 2018. Deep semantic text hashing with weak supervision. In *SIGIR*, pages 1109–1112. ACM.

Suthee Chaidaroon and Yi Fang. 2017. Variational deep semantic hashing for text documents. In *SIGIR*, pages 75–84. ACM.

Suthee Chaidaroon, Dae Hoon Park, Yi Chang, and Yi Fang. 2020. node2hash: Graph aware deep semantic text hashing. *Information Processing & Management*, 57(6):102143.

Kien Do, Truyen Tran, and Svetha Venkatesh. 2021. Clustering by maximizing mutual information across views. In *ICCV*, pages 9908–9918. IEEE.

Wei Dong, Qinliang Su, Dinghan Shen, and Changyou Chen. 2019. Document hashing with mixture-prior generative models. In *EMNLP/IJCNLP (1)*, pages 5225–5234. Association for Computational Linguistics.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. *arXiv preprint arXiv:1909.00512*.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *EMNLP (1)*, pages 6894–6910. Association for Computational Linguistics.

Tiezheng Ge, Kaiming He, Qifa Ke, and Jian Sun. 2013. Optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2946–2953. IEEE Computer Society.

Robert M. Gray and David L. Neuhoff. 1998. Quantization. *IEEE transactions on information theory*, 44(6):2325–2383.

Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. 2019. Unsupervised neural generative semantic hashing. In *SIGIR*, pages 735–744. ACM.

Casper Hansen, Christian Hansen, Jakob Grue Simonsen, Stephen Alstrup, and Christina Lioma. 2020. Unsupervised semantic hashing with pairwise reconstruction. In *SIGIR*, pages 2009–2012. ACM.

Weihua Hu, Takeru Miyato, Seiya Tokui, Eiichi Matsumoto, and Masashi Sugiyama. 2017. Learning discrete representations via information maximizing self-augmented training. In *ICML*, volume 70 of *Proceedings of Machine Learning Research*, pages 1558–1567. PMLR.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *ICLR (Poster)*. OpenReview.net.

Young Kyun Jang and Nam Ik Cho. 2021. Self-supervised product quantization for deep unsupervised image retrieval. In *ICCV*, pages 12065–12074. IEEE.

Hervé Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product quantization for nearest neighbor search. *IEEE Trans. Pattern Anal. Mach. Intell.*, 33(1):117–128.

Xu Ji, Andrea Vedaldi, and João F. Henriques. 2019. Invariant information clustering for unsupervised image classification and segmentation. In *ICCV*, pages 9864–9873. IEEE.

Yannis Kalantidis and Yannis Avrithis. 2014. Locally optimized product quantization for approximate nearest neighbor search. In *CVPR*, pages 2329–2336. IEEE Computer Society.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR (Poster)*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *ICLR*.

Harold W Kuhn. 1955. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97.

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N. Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick van Kleef, Sören Auer, and Christian Bizer. 2015. Dbpedia - A large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*, 6(2):167–195.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020. On the sentence embeddings from pre-trained language models. In *EMNLP (1)*, pages 9119–9130. Association for Computational Linguistics.

Zijing Ou, Qinliang Su, Jianxing Yu, Bang Liu, Jingwen Wang, Ruihui Zhao, Changyou Chen, and Yefeng Zheng. 2021a. Integrating semantics and neighborhood information with graph-driven generative models for document retrieval. In *ACL/IJCNLP (1)*, pages 2238–2249. Association for Computational Linguistics.

Zijing Ou, Qinliang Su, Jianxing Yu, Ruihui Zhao, Yefeng Zheng, and Bang Liu. 2021b. Refining BERT embeddings for document hashing via mutual information maximization. In *EMNLP (Findings)*, pages 2360–2369. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *EMNLP/IJCNLP (1)*, pages 3980–3990. Association for Computational Linguistics.

Ruslan Salakhutdinov and Geoffrey E. Hinton. 2009. Semantic hashing. *Int. J. Approx. Reason.*, 50(7):969–978.

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 *ACL (1)*, pages 2041–2050. Association for Computational Linguistics.

Karl Stratos and Sam Wiseman. 2020. Learning discrete structured representations by adversarially maximizing mutual information. In *ICML*, volume 119 of *Proceedings of Machine Learning Research*, pages 9144–9154. PMLR.

Fangbo Tao, Chao Zhang, Xiusi Chen, Meng Jiang, Tim Hanratty, Lance M. Kaplan, and Jiawei Han. 2018. Doc2cube: Allocating documents to text cube without labeled data. In *ICDM*, pages 1260–1265. IEEE Computer Society.

Jinpeng Wang, Ziyun Zeng, Bin Chen, Tao Dai, and Shu-Tao Xia. 2021. Contrastive quantization with code memory for unsupervised image retrieval. *CoRR*, abs/2109.05205.

Yair Weiss, Antonio Torralba, and Robert Fergus. 2008. Spectral hashing. In *NIPS*, pages 1753–1760. Curran Associates, Inc.

Shitao Xiao, Zheng Liu, Yingxia Shao, Defu Lian, and Xing Xie. 2021. Matching-oriented embedding quantization for ad-hoc retrieval. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8119–8129.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2020. Unsupervised few-bits semantic hashing with implicit topics modeling. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, pages 2566–2575. Association for Computational Linguistics.

Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *NIPS*, pages 649–657.

Lin Zheng, Qinliang Su, Dinghan Shen, and Changyou Chen. 2020. Generative semantic hashing enhanced via boltzmann machines. In *ACL*, pages 777–788. Association for Computational Linguistics.

A Details about Datasets

Table 4: Statistics of Three Benchmark Datasets.

Dataset	Classes	AvgLen	Train	Val	Test
NYT	26	648	9,221	1,154	1,152
AGNews	4	32	114,839	6,381	6,380
DBpedia	14	47	50,000	5,000	5,000

Three datasets are used to evaluate the performance of the proposed model. *1) NYT* (Tao et al., 2018) is a dataset which contains news articles published by The New York Times; *3) AGNews* (Zhang et al., 2015) is a news collection gathered from academic news search engines; *3) DBpedia* (Lehmann et al., 2015) is a dataset which contains the abstracts of articles from Wikipeida. We simply apply the string cleaning operation same as in (Ou et al., 2021b). The statistics of the three datasets are shown in Table 4.

B Retrieval

For testing, we first encode all the documents in the search set (i.e., the training set in our setting). For each document x_i in the search set, we encode it using the hard quantization operation to obtain the codeword index $\{k_i^1, k_i^2, \cdots, k_i^M\}$. Then all these codeword indices are concatenated and stored in the form of the $M \log_2 K$-bits binary code.

In terms of the retrieval stage, when given a query document x_q, we first extract its refined embedding $\tilde{z}(x_q)$ through the encoder network and then slice it into M equal-length segments as $\tilde{z}(x_q) = [\tilde{z}^1(x_q), \tilde{z}^1(x_q), \cdots, \tilde{z}^M(x_q)]$. Then we compute the Asymmetric distance (AD) (Jégou et al., 2011) between the query x_q and the documents x_i with the squared Euclidean distance metric as:

$$AD(x_q, x_i) = \sum_{m=1}^{M} ||\tilde{z}^m(x_q) - C^m \cdot h^m(x_i)||_2^2, \tag{22}$$

where $h^m(x_i) = C^m \cdot one\_hot(k_i^m)$ represents the quantized representation (i.e., one of codewords) of x_i in the m-th codebook. To compute that, we can first pre-compute a query-specific distance look-up table of size $M \times K$ that stores the distance between the segment $\tilde{z}^m(x_q)$ and all codewords in each codebook. With the pre-computed look-up table, $AD(x_q, x_i)$ can be efficiently computed by summing up the chosen values from the look-up table. It is only slightly more costly when compared with the efficiency of the Hamming distance.

C Pooling methods

Table 5: The performance comparision between the [CLS] representations and the average embeddings of BERT.

CLS vs Avg		16bits	32bits	64bits	128bits
NYT	CSH_{Avg}	71.89	71.99	77.38	78.26
	CSH_{CLS}	79.63	81.05	81.08	81.28
	$\text{MICPQ}_{\text{Avg}}$	79.56	79.99	78.49	80.33
	$\text{MICPQ}_{\text{CLS}}$	**83.15**	**84.02**	**84.24**	**85.08**
AGNews	CSH_{Avg}	81.38	81.22	81.83	81.85
	CSH_{CLS}	79.14	80.25	81.14	81.78
	$\text{MICPQ}_{\text{Avg}}$	**81.60**	**83.00**	**83.85**	**84.04**
	$\text{MICPQ}_{\text{CLS}}$	80.36	81.84	82.93	83.29
DBpedia	CSH_{Avg}	93.39	93.98	94.56	94.66
	CSH_{CLS}	95.19	95.84	95.79	96.01
	$\text{MICPQ}_{\text{Avg}}$	94.47	95.87	95.76	95.71
	$\text{MICPQ}_{\text{CLS}}$	**96.51**	**97.00**	**97.03**	**97.21**

We are interested in if taking the average embeddings of the last layers in BERT as inputs will lead to a better performance than [CLS] or not. The experiments are conducted in our proposed model MICPQ and the simple baseline CSH. Table 5 shows that these two settings achieve better performance in different datasets respectively. For simplicity, we consistently use the [CLS] representation in our experiments.