Learning Compressed Sentence Representations for On-Device Text Processing

Dinghan Shen[1*], **Pengyu Cheng**[1*], **Dhanasekar Sundararaman**[1]
Xinyuan Zhang[1], **Qian Yang**[1], **Meng Tang**[3], **Asli Celikyilmaz**[2], **Lawrence Carin**[1]
[1] Duke University [2] Microsoft Research [3] Stanford University
dinghan.shen@duke.edu

Abstract

Vector representations of sentences, trained on massive text corpora, are widely used as generic sentence embeddings across a variety of NLP problems. The learned representations are generally assumed to be *continuous* and *real-valued*, giving rise to a large memory footprint and slow retrieval speed, which hinders their applicability to low-resource (memory and computation) platforms, such as mobile devices. In this paper, we propose four different strategies to transform *continuous* and generic sentence embeddings into a *binarized* form, while preserving their rich semantic information. The introduced methods are evaluated across a wide range of downstream tasks, where the binarized sentence embeddings are demonstrated to degrade performance by only about 2% relative to their continuous counterparts, while reducing the storage requirement by over 98%. Moreover, with the learned binary representations, the semantic relatedness of two sentences can be evaluated by simply calculating their Hamming distance, which is more computational efficient compared with the inner product operation between continuous embeddings. Detailed analysis and case study further validate the effectiveness of proposed methods.

把连续的实值句子嵌入转化成二进制的形式.

在性能下降不多的情况下，降低大量的资源(内存和算力)消耗.

有多少位不同.

1 Introduction

Learning general-purpose sentence representations from large training corpora has received widespread attention in recent years. The learned sentence embeddings can encapsulate rich prior knowledge of natural language, which has been demonstrated to facilitate a variety of downstream tasks (*without* fine-tuning the encoder weights). The generic sentence embeddings can be trained either in an unsupervised manner (Kiros et al., 2015; Hill et al., 2016; Jernite et al., 2017; Gan et al., 2017; Logeswaran and Lee, 2018; Pagliardini et al., 2018), or with supervised tasks such as paraphrase identification (Wieting et al., 2016), natural language inference (Conneau et al., 2017), discourse relation classification (Nie et al., 2017), machine translation (Wieting and Gimpel, 2018), *etc*.

Significant effort has been devoted to designing better training objectives for learning sentence embeddings. However, prior methods typically assume that the general-purpose sentence representations are *continuous* and *real-valued*. However, this assumption is sub-optimal from the following perspectives: *i)* the sentence embeddings require large storage or memory footprint; *ii)* it is computationally expensive to retrieve semantically-similar sentences, since every sentence representation in the database needs to be compared, and the inner product operation is computationally involved. These two disadvantages hinder the applicability of generic sentence representations to mobile devices, where a relatively tiny memory footprint and low computational capacity are typically available (Ravi and Kozareva, 2018).

连续实值向量的缺点:
①.存储耗费内存大
②.计算消耗资源大.

In this paper, we aim to mitigate the above issues by binarizing the continuous sentence embeddings. Consequently, the embeddings require much smaller footprint, and similar sentences can be obtained by simply selecting those with closest binary codes in the Hamming space (Kiros and Chan, 2018). One simple idea is to naively binarize the continuous vectors by setting a hard threshold. However, we find that this strategy leads to significant performance drop in the empirical results. Besides, the dimension of the binary sentence embeddings cannot be flexibly chosen with this strategy, further limiting the practice use of the direct binarization method.

In this regard, we propose three alternative strategies to parametrize the transformation from

共提出了4种二进制化方法

[*] Equal contribution.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 107–116
Florence, Italy, July 28 - August 2, 2019.

pre-trained generic continuous embeddings to their binary forms. Our exploration spans from *simple* operations, such as a random projection, to *deep* neural network models, such as a *regularized* autoencoder. Particularly, we introduce a semantic-preserving objective, which is augmented with the standard autoenoder architecture to encourage abstracting informative binary codes. InferSent (Conneau et al., 2017) is employed as the testbed sentence embeddings in our experiments, but the binarization schemes proposed here can easily be extended to other pre-trained general-purpose sentence embeddings. We evaluate the quality of the learned general-purpose binary representations using the SentEval toolkit (Conneau et al., 2017). It is observed that the inferred binary codes successfully maintain the semantic features contained in the continuous embeddings, and only lead to around 2% performance drop on a set of downstream NLP tasks, while requiring merely 1.5% memory footprint of their continuous counterparts.

Moreover, on several sentence matching benchmarks, we demonstrate that the relatedness between a sentence pair can be evaluated by simply calculating the Hamming distance between their binary codes, which perform on par with or even superior than measuring the cosine similarity between continuous embeddings (see Table 1). Note that computing the Hamming distance is much more computationally efficient than the inner product operation in a continuous space. We further perform a K-nearest neighbor sentence retrieval experiment on the SNLI dataset (Bowman et al., 2015), and show that those semantically-similar sentences can indeed be efficiently retrieved with off-the-shelf binary sentence representations. Summarizing, our contributions in this paper are as follows:

i) to the best of our knowledge, we conduct the first systematic exploration on learning general-purpose *binarized* (memory-efficient) sentence representations, and four different strategies are proposed;

ii) an autoencoder architecture with a carefully-designed *semantic-preserving loss* exhibits strong empirical results on a set of downstream NLP tasks;

iii) more importantly, we demonstrate, on several sentence-matching datasets, that simply evaluating the *Hamming distance* over binary representations performs on par or even better than calculating the *cosine similarity* between their continuous counterparts (which is less computationally-efficient).

2 Related Work

Sentence representations pre-trained from a large amount of data have been shown to be effective when transferred to a wide range of downstream tasks. Prior work along this line can be roughly divided into two categories: *i*) pre-trained models that require fine-tuning on the specific transferring task (Dai and Le, 2015; Ruder and Howard, 2018; Radford et al., 2018; Devlin et al., 2018; Cer et al., 2018); *ii*) methods that extract general-purpose sentence embeddings, which can be effectively applied to downstream NLP tasks *without* fine-tuning the encoder parameters (Kiros et al., 2015; Hill et al., 2016; Jernite et al., 2017; Gan et al., 2017; Adi et al., 2017; Logeswaran and Lee, 2018; Pagliardini et al., 2018; Tang and de Sa, 2018). Our proposed methods belong to the second category and provide a generic and easy-to-use encoder to extract highly informative sentence representations. However, our work is unique since the embeddings inferred from our models are binarized and compact, and thus possess the advantages of small memory footprint and much faster sentence retrieval.

Learning memory-efficient embeddings with deep neural networks has attracted substantial attention recently. One general strategy towards this goal is to extract discrete or binary data representations (Jang et al., 2016; Shu and Nakayama, 2017; Dai et al., 2017; Chen et al., 2018; Shen et al., 2018; Tissier et al., 2019). Binarized embeddings are especially attractive because they are more memory-efficient (relative to discrete embeddings), and they also enjoy the advantages of fast retrieval based upon a Hamming distance calculation. Previous work along this line in NLP has mainly focused on learning compact representations at the word-level (Shu and Nakayama, 2017; Chen et al., 2018; Tissier et al., 2019), while much less effort has been devoted to extracting binarized embeddings at the sentence-level. Our work aims to bridge this gap, and serves as an initial attempt to facilitate the deployment of state-of-the-art sentence embeddings on on-device mobile applications.

Our work is also related to prior research on semantic hashing, which aims to learn binary text embeddings specifically for the information retrieval task (Salakhutdinov and Hinton, 2009; Zhang et al., 2010; Wang et al., 2014; Xu et al., 2015; Shen et al., 2018). However, these methods are typically trained and evaluated on documents that belong to a specific domain, and thus cannot serve as generic binary sentence representation applicable to a wide variety of NLP taks. In contrast, our model is trained on large corpora and seeks to provide general-purpose binary representations that can be leveraged for various application scenarios.

3 Proposed Approach

We aim to produce compact and binarized representations from continuous sentence embeddings, and preserve the associated semantic information. Let x and f denote, respectively, an input sentence and the function defined by a pre-trained general-purpose sentence encoder. Thus, $f(x)$ represents the continuous embeddings extracted by the encoder. The goal of our model is to learn a universal transformation g that can convert $f(x)$ to highly informative binary sentence representations, *i.e.*, $g(f(x))$, which can be used as generic features for a collection of downstream tasks. We explore four strategies to parametrize the transformation g.

二进制转换函数

句子编码函数

句子

3.1 Hard Threshold

方法

We use h and b to denote the continuous and binary sentence embeddings, respectively, and L denotes the dimension of h. The first method to binarize the continuous representations is to simply convert each dimension to either 0 or 1 based on a hard threshold. This strategy requires no training and directly operates on pre-trained continuous embeddings. Suppose s is the hard threshold, we have, for $i = 1, 2, ..., L$:

$$b^{(i)} = \mathbf{1}_{h^{(i)} > s} = \frac{\text{sign}(h^{(i)} - s) + 1}{2}, \qquad (1)$$

缺点

One potential issue of this direct binarization method is that the information contained in the continuous representations may be largely lost, since there is no training objective encouraging the preservation of semantic information in the produced binary codes (Shen et al., 2018). Another disadvantage is that the length of the resulting binary code must be the same as the original continuous representation, and can not be flexibly chosen. In practice, however, we may want to learn shorter binary embeddings to save more memory footprint or computation.

3.2 Random Projection

To tackle the limitation of the above direct binarization method, we consider an alternative strategy that requires no training either: simply applying a random projection over the pre-trained continuous representations. Wieting and Kiela (2018) has shown that random sentence encoders can effectively construct universal sentence embeddings from word vectors, while possessing the flexibility of adaptively altering the embedding dimensions. Here, we are interested in exploring whether a random projection would also work well while transforming continuous sentence representations into their binary counterparts.

二进制向量维度

We randomly initialize a matrix $W \in \mathbb{R}^{D \times L}$, 连续实值向量维度

where D denotes the dimension of the resulting binary representations. Inspired by the standard initialization heuristic employed in (Glorot and Bengio, 2010; Wieting and Kiela, 2018), the values of the matrix are initialized as sampled uniformly. For $i = 1, 2, \ldots, D$ and $j = 1, 2, \ldots, L$, we have:

$$\boxed{W_{i,j} \sim \text{Uniform}(-\frac{1}{\sqrt{D}}, \frac{1}{\sqrt{D}}),} \qquad (2)$$

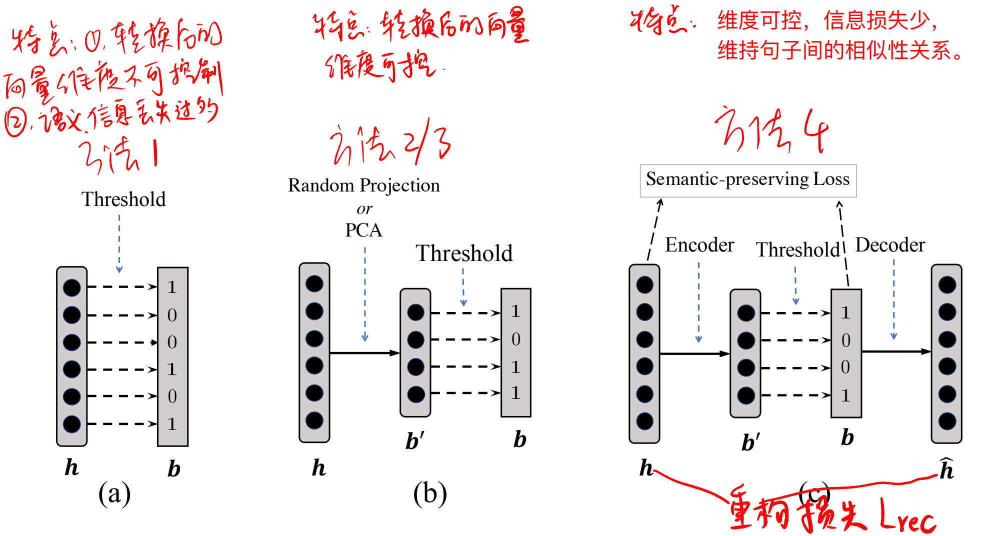
先映射到想要的维度后再进行Hard Threshold变换.

After converting the continuous sentence embeddings to the desired dimension D with the matrix randomly initialized above, we further apply the operation in (1) to binarize it into the discrete/compact form. The dimension D can be set arbitrarily with this approach, which is easily applicable to any pre-trained sentence embeddings (since no training is needed). This strategy is related to the Locality-Sensitive Hashing (LSH) for inferring binary embeddings (Van Durme and Lall, 2010).

3.3 Principal Component Analysis

We also consider an alternative strategy to adaptively choose the dimension of the resulting binary representations. Specifically, Principal Component Analysis (PCA) is utilized to reduce the dimensionality of pre-trained continuous embeddings.

Given a set of sentences $\{x_i\}_{i=1}^N$ and their corresponding continuous embeddings $\{h_i\}_{i=1}^N \subset \mathbb{R}^L$, we learn a projection matrix to reduce the embedding dimensions while keeping the embeddings distinct as much as possible. After centralizing the embeddings as $h_i = h_i - \frac{1}{N}\sum_{i=1}^N h_i$, the data, as



特点：①. 转换后的向量维度不可控制 ②. 语义信息丢失过多

特点：转换后的向量维度可控。

特点：维度可控，信息损失少，维持句子间的相似性关系。

Figure 1: Proposed model architectures: (a) direct binarization with a hard threshold s; (b) reducing the dimensionality with either a random projection or PCA, followed by a binarization step; (c) an encoding-decoding framework with an additional semantic-preserving loss.

a matrix $H = (h_1, h_2, \ldots, h_N)$, has the singular value decomposition (SVD):

通过PCA，把L维的连续实值向量映射到D维。再进行hard threshold转换

$$\underbrace{H}_{L\times N} = \underbrace{U}_{L\times L}\underbrace{\Lambda}_{L\times N}\boxed{V^T}^{N\times N},$$

where Λ is an $L \times N$ matrix with descending singular values of X on its diagonal, with U and V orthogonal matrices. Then the correlation matrix can be written as: $HH^T = U\Lambda^2U^T$. Assume that the diagonal matrix $\Lambda^2 = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_L)$ has descending elements $\lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_L \geq 0$. We select first D rows of U as our projection matrix $W = U_{1:D}$, then the correlation matrix of WH is $WHH^TW^T = \mathrm{diag}(\lambda_1, \lambda_2, \ldots, \lambda_D)$, which indicates that the embeddings are projected to D independent and most distinctive axes.

After projecting continuous embeddings to a representative lower dimensional space, we apply the hard threshold function at the position 0 to obtain the binary representations (since the embeddings are zero-centered).

3.4 Autoencoder Architecture ☆

以上3种方法的缺点

The methods proposed above suffer from the common issue that the model objective does not explicitly encourage the learned binary codes to retain the semantic information of the original continuous embeddings, and a separate binarization step is employed after training. To address this shortcoming, we further consider an autoencoder architecture, that leverages the reconstruction loss to hopefully endow the learned binary representations with more information. Specifically, an encoder network is utilized to transform the continuous into a binary latent vector, which is then reconstructed back with a decoder network.

第4种方法

For the encoder network, we use a matrix operation, followed by a binarization step, to extract useful features (similar to the random projection setup). Thus, for $i = 1, 2, \ldots, D$, we have:

encoder

$$\begin{aligned} b^{(i)} &= \mathbf{1}_{\sigma(W_i \cdot h + k^{(i)}) > s^{(i)}} \\ &= \frac{\mathrm{sign}(\sigma(W_i \cdot h + k^{(i)}) - s^{(i)}) + 1}{2}, \end{aligned} \tag{3}$$

where k is the bias term and $k^{(i)}$ corresponds to the i-th element of k. $s^{(i)}$ denotes the threshold determining whether the i-th bit is 0 or 1. During training, we may use either *deterministic* or *stochastic* binarization upon the latent variable. For the deterministic case, $s^{(i)} = 0.5$ for all dimensions; in the stochastic case, $s^{(i)}$ is uniformly sampled as: $s^{(i)} \sim \mathrm{Uniform}(0, 1)$. We conduct an empirical comparison between these two binarization strategies in Section 4.

确定性的 随机的

$k^{(i)}$ 和 $s^{(i)}$ 的设置方法

Prior work have shown that linear decoders are favorable for learning binary codes under the encoder-decoder framework (Carreira-Perpinán and Raziperchikolaei, 2015; Dai et al., 2017; Shen et al., 2018). Inspired by these results, we employ a linear transformation to reconstruct the original continuous embeddings from the binary codes:

decoder

$$\hat{h}^{(i)} = W_i' \cdot b + k'^{(i)}, \tag{4}$$

where W' and k' are weight and bias term respectively, which are learned. The mean square error between h and $\hat{h}$ is employed as the reconstruction loss:

重构损失

$$\mathcal{L}_{rec} = \frac{1}{D}\sum_{i=1}^{D}(h^{(i)} - \hat{h}^{(i)})^2, \tag{5}$$

This objective imposes the binary vector b to encode more information from h (leading to smaller reconstruction error). Straight-through (ST) estimator (Hinton, 2012) is utilized to estimate the gradients for the binary variable. The autoencoder model is optimized by minimizing the reconstruction loss for all sentences. After training, the encoder network is leveraged as the transformation to convert the pre-trained continuous embeddings into the binary form.

3.4.1 Semantic-preserving Regularizer

只使用重构误差的不足

Although the reconstruction objective can help the binary variable to endow with richer semantics, there is no loss that explicitly encourages the binary vectors to preserve the similarity information contained in the original continuous embeddings. Consequently, the model may lead to small reconstruction error but yield sub-optimal binary representations (Tissier et al., 2019). To improve the semantic-preserving property of the inferred binary embeddings, we introduce an additional objective term.

Consider a triple group of sentences $(x_\alpha, x_\beta, x_\gamma)$, whose continuous embeddings are $(h_\alpha, h_\beta, h_\gamma)$, respectively. Suppose that the cosine similarity between h_α and h_β is larger than that between h_β and h_γ, then it is desirable that the Hamming distance between b_α and b_β should be smaller than that between b_β and b_γ (notably, both large cosine similarity and small Hamming distance indicate that two sentences are semantically-similar).

Let $d_c(\cdot, \cdot)$ and $d_h(\cdot, \cdot)$ denote the cosine similarity and Hamming distance (in the continuous and binary embedding space), respectively. Define $l_{\alpha,\beta,\gamma}$ as an indicator such that, $l_{\alpha,\beta,\gamma} = 1$ if $d_c(h_\alpha, h_\beta) \geq d_c(h_\beta, h_\gamma)$, and $l_{\alpha,\beta,\gamma} = -1$ otherwise. The semantic-preserving regularizer is then defined as:

$$\begin{cases} 1 & 若\ \alpha,\beta的相似度 > \beta.\gamma \\ -1 & 反之 \end{cases}$$

$$\mathcal{L}_{sp} = \sum_{\alpha,\beta,\gamma} \max\{0, l_{\alpha,\beta,\gamma}[d_h(b_\alpha, b_\beta) - d_h(b_\beta, b_\gamma)]\}, \quad (6)$$

两者的汉明距离，
越↑，越不相似.

By penalizing L_{sp}, the learned transformation function g is explicitly encouraged to retain the semantic similarity information of the original continuous embeddings. Thus, the entire objective function to be optimized is:

$$\mathcal{L} = \mathcal{L}_{rec} + \lambda_{sp}\mathcal{L}_{sp}, \quad (7)$$

where λ_{sp} controls the relative weight between the reconstruction loss ($\mathcal{L}_{rec}$) and semantic-preserving loss ($\mathcal{L}_{sp}$).

3.5 Discussion

Another possible strategy is to directly train the general-purpose binary embeddings from scratch, *i.e.*, jointly optimizing the continuous embeddings training objective and continuous-to-binary parameterization. However, our initial attempts demonstrate that this strategy leads to inferior empirical results. This observation is consistent with the results reported in (Kiros and Chan, 2018). Specifically, a binarization layer is directly appended over the InferSent architecture (Conneau et al., 2017) during training, which gives rise to much larger drop in terms of the embeddings' quality (we have conducted empirical comparisons with (Kiros and Chan, 2018) in Table 1). Therefore, here we focus on learning universal binary embeddings based on pretained continuous sentence representations.

从实验结果看
这种方法的
性能不好.

4 Experimental setup

4.1 Pre-trained Continuous Embeddings

Our proposed model aims to produce highly informative binary sentence embeddings based upon pre-trained continuous representations. In this paper, we utilize InferSent (Conneau et al., 2017) as the continuous embeddings (given its effectiveness and widespread use). Note that all four proposed strategies can be easily extended to other pre-trained general-purpose sentence embeddings as well.

Specifically, a bidirectional LSTM architecture along with a max-pooling operation over the hidden units is employed as the sentence encoder, and the model parameters are optimized on the natural language inference tasks, *i.e.*, Standford Natural Language Inference (SNLI) (Bowman et al., 2015) and Multi-Genre Natural Language Inference (MultiNLI) datasets (Williams et al., 2017).

4.2 Training Details

Our model is trained using Adam (Kingma and Ba, 2014), with a value 1×10^{-5} as the learning rate for all the parameters. The number of bits (*i.e.*, dimension) of the binary representation is set as 512, 1024, 2048 or 4096, and the best choice for each model is chosen on the validation set, and the corresponding test results are presented in Table 1. The batch size is chosen as 64 for all model variants. The hyperparameter over λ_{sp} is selected from $\{0.2, 0.5, 0.8, 1\}$ on the validation set, and 0.8 is found to deliver the best empirical results.

句子分类任务 句子匹配任务

Model	Dim	MR	CR	SUBJ	MPQA	SST	STS14	STSB	SICK-R	MRPC
***Continuous** (**dense**) sentence embeddings*										
fastText-BoV	300	78.2	80.2	91.8	88.0	82.3	.65/.63	58.1/59.0	0.698	67.9/74.3
SkipThought	4800	76.5	80.1	93.6	87.1	82.0	.29/.35	41.0/41.7	0.595	57.9/66.6
SkipThought-LN	4800	79.4	83.1	**93.7**	89.3	82.9	.44/.45	-	-	-
InferSent-FF	4096	79.7	84.2	92.7	89.4	84.3	.68/.66	55.6/56.2	0.612	**67.9/73.8**
InferSent-G	4096	**81.1**	**86.3**	92.4	**90.2**	**84.6**	**.68/.65**	**70.0/68.0**	**0.719**	67.4/73.2
***Binary** (**compact**) sentence embeddings*										
InferLite-short	256	73.7	81.2	83.2	86.2	78.4	0.61/-	63.4/63.3	0.597	61.7/70.1
InferLite-medium	1024	76.3	83.2	87.8	88.4	81.3	0.67/-	64.9/64.9	0.642	64.1/72.0
InferLite-long	4096	77.7	83.7	89.6	89.1	82.3	0.68/-	67.9/67.6	0.663	65.4/**72.9**
HT-binary	4096	76.6	79.9	**91.0**	88.4	80.6	.62/.60	55.8/53.6	0.652	65.6/70.4
Rand-binary	2048	78.7	82.7	90.4	88.9	81.3	.66/.63	65.1/62.3	0.704	65.7/70.8
PCA-binary	2048	78.4	84.5	90.7	89.4	81.0	.66/.65	63.7/62.8	0.518	65.0/ 69.7
AE-binary	2048	78.7	**84.9**	90.6	89.6	82.1	.68/.66	71.7/69.7	0.673	65.8/70.8
AE-binary-SP	2048	**79.1**	84.6	90.8	**90.0**	**82.7**	**.69/.67**	**73.2/70.6**	**0.705**	**67.2**/72.0

Inferlite的二进制版本

以Infersent为预训练的连续实值向量

结果 SP loss

Table 1: Performance on the test set for 10 downstream tasks. The STS14, STSB and MRPC are evaluated with Pearson and Spearman correlations, and SICK-R is measured with Pearson correlation. All other datasets are evaluated with test accuracy. InferSent-G uses Glove (G) as the word embeddings, while InferSent-FF employs FastText (F) embeddings with Fixed (F) padding. The empirical results of InferLite with different lengths of binary embeddings, *i.e.*, 256, 1024 and 4096, are considered.

The training with the autoencoder setup takes only about 1 hour to converge, and thus can be readily applicable to even larger datasets.

4.3 Evaluation

To facilitate comparisons with other baseline methods, we use SentEval toolkit[1] (Conneau and Kiela, 2018) to evaluate the learned binary (compact) sentence embeddings. Concretely, the learned representations are tested on a series of downstream tasks to assess their transferability (with the encoder weights fixed), which can be categorized as follows:

- **Sentence classification**, including sentiment analysis (MR, SST), product reviews (CR), subjectivity classification (SUBJ), opinion polarity detection (MPQA) and question type classification (TREC). A linear classifier is trained with the generic sentence embeddings as the input features. The default SentEval settings is used for all the datasets.

- **Sentence matching**, which comprises semantic relatedness (SICK-R, STS14, STSB) and paraphrase detection (MRPC). Particularly, each pair of sentences in STS14 dataset is associated with a similarity score from 0 to 5 (as the corresponding label). Hamming distance between the binary representations is directly leveraged as the prediction score (*without* any classifier parameters).

For the sentence matching benchmarks, to allow fair comparison with the continuous embeddings, we do not use the same classifier architecture in SentEval. Instead, we obtain the predicted relatedness by directly computing the *cosine similarity* between the continuous embeddings. Consequently, there are no classifier parameters for both the binary and continuous representations. The same valuation metrics in SentEval(Conneau and Kiela, 2018) are utilized for all the tasks. For MRPC, the predictions are made by simply judging whether a sentence pair's score is larger or smaller than the averaged Hamming distance (or cosine similarity).

4.4 Baselines

We consider several strong baselines to compare with the proposed methods, which include both **continuous** (dense) and **binary** (compact) representations. For the continuous generic sentence embeddings, we make comparisons with fastText-BoV (Joulin et al., 2016), Skip-Thought Vectors (Kiros et al., 2015) and InferSent (Conneau et al., 2017). As to the binary embeddings, we consider the binarized version of InferLite (Kiros and Chan, 2018), which, as far as we are concerned, is the only general-purpose binary representations baseline reported.

5 Experimental Results

We experimented with five model variants to learn general-purpose binary embeddings: *HT-binary* (hard threshold, which is selected from

[1] https://github.com/facebookresearch/SentEval

$\{0, 0.01, 0.1\}$ on the validation set), *Rand-binary* (random projection), *PCA-binary* (reduce the dimensionality with principal component analysis), *AE-binary* (autoencoder with the reconstruction objective) and *AE-binary-SP* (autoencoder with both the reconstruction objective and Semantic-Preserving loss). Our code will be released to encourage future research.

5.1 Task transfer evaluation

We evalaute the binary sentence representations produced by different methods with a set of transferring tasks. The results are shown in Table 1. The proposed autoencoder architecture generally demonstrates the best results. Especially while combined with the semantic-preserving loss defined in (7), AE-binary-SP exhibits higher performance compared with a standard autoencoder. It is worth noting that the Rand-binary and PCA-binary model variants also show competitive performance despite their simplicity. These strategies are also quite promising given that no training is required given the pre-trained continuous sentence representations.

Another important result is that, the AE-binary-SP achieves competitive results relative to the InferSent, leading to only about 2% loss on most datasets and even performing at par with InferSent on several datasets, such as the MPQA and STS14 datasets. On the sentence matching tasks, the yielded binary codes are evaluated by merely utilizing the hamming distance features (as mentioned above). To allow fair comparison, we compare the predicted scores with the cosine similarity scores based upon the continuous representations (there are no additional parameters for the classifier). The binary codes brings out promising empirical results relative to their continuous counterparts, and even slightly outperform InferSent on the STS14 dataset.

We also found that our AE-binary-SP model variant consistently demonstrate superior results than the InferLite baselines, which optimize the NLI objective directly over the binary representations. This may be attributed to the difficulty of backpropagating gradients through discrete/binary variables, and would be an interesting direction for future research.

5.2 Nearest Neighbor Retrieval

Case Study One major advantage of binary sentence representations is that the similarity of two sentences can be evaluated by merely calculating the hamming distance between their binary codes. To gain more intuition regarding the semantic information encoded in the binary embeddings, we convert all the sentences in the SNLI dataset into continuous and binary vectors (with InferSent-G and AE-binary-SP, respectively). The top-3 closet sentences are retrieved based upon the corresponding metrics, and the resulting samples are shown in Table 2. It can be observed that the sentences selected based upon the Hamming distance indeed convey very similar semantic meanings. In some cases, the results with binary codes are even more reasonable compared with the continuous embeddings. For example, for the first query, all three sentences in the left column relate to "watching a movie", while one of the sentences in the right column is about "sleeping".

Retrieval Speed The bitwise comparison is much faster than the element-wise multiplication operation (between real-valued vectors) (Tissier et al., 2019). To verify the speed improvement, we sample 10000 sentence pairs from SNLI and extract their continuous and binary embeddings (with the same dimension of 4096), respectively. We record the time to compute the cosine similarity and hamming distance between the corresponding representations. With our Python implementation, it takes 3.67μs and 288ns respectively, indicating that calculating the Hamming distance is over 12 times faster. Our implementation is not optimized, and the running time of computing Hamming distance can be further improved (to be proportional to the number of different bits, rather than the input length[2]).

5.3 Ablation Study

5.3.1 The effect of semantic-preserving loss

To investigate the importance of incorporating the locality-sensitive regularizer, we select different values of λ_{sp} (ranging from 0.0 to 1.0) and explore how the transfer results would change accordingly. The λ_{sp} controls the relative weight of the semantic-preserving loss term. As shown in Table 3, augmenting the semantic-preserving loss consistently improves the quality of learned binary embeddings, while the best test accuracy on the MR dataset is obtained with $\lambda_{sp} = 0.8$.

[2] https://en.wikipedia.org/wiki/Hamming_distance

Hamming Distance (binary embeddings)	Cosine Similarity (continuous embeddings)
Query: Several people are sitting in a movie theater .	
A group of people watching a movie at a theater .	A group of people watching a movie at a theater .
A crowd of people are watching a movie indoors .	A man is watching a movie in a theater .
A man is watching a movie in a theater .	Some people are sleeping on a sofa in front of the television .
Query: A woman crossing a busy downtown street .	
A lady is walking down a busy street .	A woman walking on the street downtown .
A woman is on a crowded street .	A lady is walking down a busy street .
A woman walking on the street downtown .	A man and woman walking down a busy street .
Query: A well dressed man standing in front of piece of artwork .	
A well dressed man standing in front of an abstract fence painting .	A man wearing headphones is standing in front of a poster .
A man wearing headphones is standing in front of a poster .	A man standing in front of a chalkboard points at a drawing .
A man in a blue shirt standing in front of a garage-like structure painted with geometric designs .	A man in a blue shirt standing in front of a garage-like structure painted with geometric designs .
Query: A woman is sitting at a bar eating a hamburger .	
A woman sitting eating a sandwich .	A woman is sitting in a cafe eating lunch .
A woman is sitting in a cafe eating lunch .	A woman is eating at a diner .
The woman is eating a hotdog in the middle of her bedroom .	A woman is eating her meal at a resturant .
Query: Group of men trying to catch fish with a fishing net .	
Two men are on a boat trying to fish for food during a sunset .	There are three men on a fishing boat trying to catch bass .
There are three men on a fishing boat trying to catch bass .	Two men are trying to fish .
Two men pull a fishing net up into their red boat .	Two men are on a boat trying to fish for food during a sunset .

Table 2: Nearest neighbor retrieval results on the SNLI dataset. Given a a query sentence, the left column shows the top-3 retrieved samples based upon the hamming distance with all sentences' binary representations, while the right column exhibits the samples according to the cosine similarity of their continuous embeddings.

λ_{sp}	0.0	0.2	0.5	0.8	1.0
Accuracy	78.2	78.5	78.5	**79.1**	78.4

Table 3: Ablation study for the AE-binary-SP model with different choices of λ_{sp} (evaluated with test accuracy on the MR dataset).

5.3.2 Sampling strategy

As discussed in Section 3.4, the binary latent vector b can be obtained with either a deterministic or stochastically sampled threshold. We compare these two sampling strategies on several downstream tasks. As illustrated in Figure 2, setting a fixed threshold demonstrates better empirical performance on all the datasets. Therefore, deterministic threshold is employed for all the autoencoder model variants in our experiments.

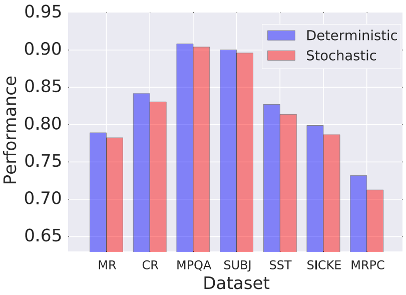


Figure 2: The comparison between *deterministic* and *stochastic* sampling for the autoencoder strategy.

5.3.3 The effect of embedding dimension

Except for the hard threshold method, other three proposed strategies all possess the flexibility of adaptively choosing the dimension of learned binary representations. To explore the sensitivity of extracted binary embeddings to their dimensions, we run four model variants (Rand-binary, PCA-binary, AE-binary, AE-binary-SP) with different number of bits (*i.e.*, 512, 1024, 2048, 4096), and their corresponding results on the MR dataset are shown in Figure 3.

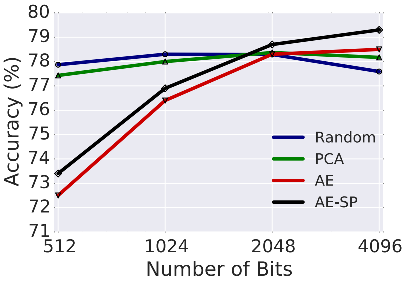


Figure 3: The test accuracy of different model on the MR dataset across 512, 1024, 2048, 4096 bits for the learned binary representations.

For the AE-binary and AE-binary-SP models, longer binary codes consistently deliver better results. While for the Rand-binary and PCA-binary variants, the quality of inferred representations is much less sensitive to the embedding dimension. Notably, these two strategies exhibit competitive performance even with only 512 bits. Therefore, in the case where less memory footprint or little training is preferred, Rand-binary and PCA-binary could be more judicious choices.

6 Conclusion

This paper presents a first step towards learning binary and general-purpose sentence representations that allow for efficient storage and fast retrieval over massive corpora. To this end, we explore four distinct strategies to convert pre-trained continuous sentence embeddings into a binarized form. Notably, a regularized autoencoder augmented with semantic-preserving loss exhibits the best empirical results, degrading performance by only around 2% while saving over 98% memory footprint. Besides, two other model variants with a random projection or PCA transformation require no training and demonstrate competitive embedding quality even with relatively small dimensions. Experiments on nearest-neighbor sentence retrieval further validate the effectiveness of proposed framework.

References

Yossi Adi, Einat Kermany, Yonatan Belinkov, Ofer Lavi, and Yoav Goldberg. 2017. Fine-grained analysis of sentence embeddings using auxiliary prediction tasks. *CoRR*, abs/1608.04207.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *EMNLP*.

Miguel A Carreira-Perpinán and Ramin Raziperchikolaei. 2015. Hashing with binary autoencoders. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 557–566.

Daniel Cer, Yinfei Yang, Sheng yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder. *CoRR*, abs/1803.11175.

Ting Chen, Martin Renqiang Min, and Yizhou Sun. 2018. Learning k-way d-dimensional discrete codes for compact embedding representations. *arXiv preprint arXiv:1806.09464*.

Alexis Conneau and Douwe Kiela. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. 2017. Supervised learning of universal sentence representations from natural language inference data. In *EMNLP*.

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in neural information processing systems*, pages 3079–3087.

Bo Dai, Ruiqi Guo, Sanjiv Kumar, Niao He, and Le Song. 2017. Stochastic generative hashing. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 913–922. JMLR. org.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Zhe Gan, Yunchen Pu, Ricardo Henao, Chunyuan Li, Xiaodong He, and Lawrence Carin. 2017. Learning generic sentence representations using convolutional neural networks. In *EMNLP*.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. In *HLT-NAACL*.

G Hinton. 2012. Neural networks for machine learning. coursera,[video lectures].

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

Yacine Jernite, Samuel R. Bowman, and David A Sontag. 2017. Discourse-based objectives for fast unsupervised sentence representation learning. *CoRR*, abs/1705.00557.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

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

Jamie Kiros and William Chan. 2018. Inferlite: Simple universal sentence representations from natural language inference data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4868–4874.

Ryan Kiros, Yukun Zhu, Ruslan Salakhutdinov, Richard S. Zemel, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2015. Skip-thought vectors. In *NIPS*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *ICLR*.

Allen Nie, Erin D. Bennett, and Noah D. Goodman. 2017. Dissent: Sentence representation learning from explicit discourse relations. *CoRR*, abs/1710.04334.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2018. Unsupervised learning of sentence embeddings using compositional n-gram features. In *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/research-covers/languageunsupervised/language understanding paper. pdf*.

Sujith Ravi and Zornitsa Kozareva. 2018. Self-governing neural networks for on-device short text classification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 804–810.

Sebastian Ruder and Jeremy Howard. 2018. Universal language model fine-tuning for text classification. In *ACL*.

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

Dinghan Shen, Qinliang Su, Paidamoyo Chapfuwa, Wenlin Wang, Guoyin Wang, Lawrence Carin, and Ricardo Henao. 2018. Nash: Toward end-to-end neural architecture for generative semantic hashing. In *ACL*.

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

Shuai Tang and Virginia R de Sa. 2018. Improving sentence representations with multi-view frameworks. *arXiv preprint arXiv:1810.01064*.

Julien Tissier, Amaury Habrard, and Christophe Gravier. 2019. Near-lossless binarization of word embeddings. *AAAI*.

Benjamin Van Durme and Ashwin Lall. 2010. Online generation of locality sensitive hash signatures. In *Proceedings of the ACL 2010 Conference Short Papers*, pages 231–235. Association for Computational Linguistics.

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

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2016. Towards universal paraphrastic sentence embeddings. *CoRR*, abs/1511.08198.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*.

John Wieting and Douwe Kiela. 2018. No training required: Exploring random encoders for sentence classification. *CoRR*, abs/1901.10444.

Adina Williams, Nikita Nangia, and Samuel R Bowman. 2017. A broad-coverage challenge corpus for sentence understanding through inference. *arXiv preprint arXiv:1704.05426*.

Jiaming Xu, Peng Wang, Guanhua Tian, Bo Xu, Jun Zhao, Fangyuan Wang, and Hongwei Hao. 2015. Convolutional neural networks for text hashing. In *Twenty-Fourth International Joint Conference on Artificial Intelligence*.

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.