Embedding Compression with Isotropic Iterative Quantization

Siyu Liao,[1] Jie Chen,[2] Yanzhi Wang,[3] Qinru Qiu,[4] Bo Yuan[1]
[1]Department of Electrical and Computer Engineering, Rutgers University
[2]MIT-IBM Watson AI Lab, IBM Research
[3]Department of Electrical and Computer Engineering, Northeastern University
[4]Electrical Engineering and Computer Science Department, Syracuse University
siyu.liao@rutgers.edu, chenjie@us.ibm.com, yanz.wang@northeastern.edu
qiqiu@syr.edu, bo.yuan@soe.rutgers.edu

Abstract

Continuous representation of words is a standard component in deep learning-based NLP models. However, representing a large vocabulary requires significant memory, which can cause problems, particularly on resource-constrained platforms. Therefore, in this paper we propose an isotropic iterative quantization (IIQ) approach for compressing embedding vectors into binary ones, leveraging the iterative quantization technique well established for image retrieval, while satisfying the desired isotropic property of PMI based models. Experiments with pre-trained embeddings (i.e., GloVe and HDC) demonstrate a more than thirty-fold compression ratio with comparable and sometimes even improved performance over the original real-valued embedding vectors.

1 Introduction

Words are basic units in many natural language processing (NLP) applications, e.g., translation (Bahdanau, Cho, and Bengio 2014) and text classification (Joulin et al. 2016). Understanding words is crucial but can be very challenging. One difficulty lies in the large vocabulary commonly seen in applications. Moreover, their semantic permutations can be numerous, constituting rich expressions at the sentence and paragraph levels.

In statistical language models, word distributions are learned for unigrams, bigrams, and generally n-grams. A unigram distribution presents the probability for each word. The histogram is already sufficiently complex given a large vocabulary. Then, the complexity of bigram distributions is quadratic in the vocabulary size and that of n-gram ones is exponential. The combinatorial nature motivates researchers to develop alternative representations which otherwise explode.

Instead of word distributions, continuous representations with floating-point vectors are much more convenient to handle: they are differentiable, and their differences can be used to draw semantic analogy. A variety of algorithms were proposed over the years for learning these word vectors. Two representative ones are Word2Vec (Mikolov et al. 2013a) and GloVe (Pennington, Socher, and Manning 2014). Word2Vec is a classical algorithm based on either skip grams or a bag of words, both of which are unsupervised and can directly learn word embeddings from a given corpus. GloVe is another embedding learning algorithm, which combines the advantage of a global factorization of the word co-occurrence matrix, as well as that of the local context. Both approaches are effective in many NLP applications, including word analogy and name entity recognition.

Neural networks with word embeddings are frequently used in solving NLP problems, such as sentiment analysis (dos Santos and Gatti 2014) and name entity recognition (Lample et al. 2016). An advantage of word embeddings is that interactions between words may be modeled by using neural network layers (e.g., attention architectures).

Despite the success of these word embeddings, they often constitute a substantial portion of the overall model. For example, the pre-trained Word2Vec (Mikolov et al. 2013b) contains 3M word vectors and the storage is approximately 3GB. This cost becomes a bottleneck in deployment on resource-constrained platforms.

Thus, much work studies the compression of word embeddings. (Shu and Nakayama 2017) propose to represent word vectors by using multiple codebooks trained with Gumbel-softmax. (Grzegorczyk and Kurdziel 2017) learn binary document emebddings via a bag-of-word-like process. The learned vectors are demonstrated to be effective for document retrieval.

In information retrieval, iterative quantization (ITQ) (Gong et al. 2013) transforms vectors into binary ones, which are found to be successful in image retrieval. The method maximizes the bit variance meanwhile minimizing the quantization loss. It is theoretically sound and also computationally efficient. However, (Grzegorczyk and Kurdziel 2017) find that directly applying ITQ in NLP tasks may not be effective.

In (Mu, Bhat, and Viswanath 2017), authors propose an alternate approach that improves the quality of word embeddings without incurring extra training. The main idea lies in the concept of isotropy used to explain the success of pointwise mutual information (PMI) based embeddings. The authors demonstrate that the isotropy could be improved

through projecting embedding vectors toward weak directions.

Therefore, in this work we propose *isotropic iterative quantization* (IIQ), which leverages iterative quantization meanwhile satisfying the isotropic property. The main idea is to optimize a new objective function regarding the isotropy of word embeddings, rather than maximizing the bit variance.

Maximizing the bit variance and maximizing isotropy are two opposite ideas, because the former performs projection toward large eigenvalues (dominant directions) while the latter projects toward the smallest ones (weak directions). Given prior success (Mu, Bhat, and Viswanath 2017), it is argued that maximizing isotropy is more beneficial in NLP applications.

2 Related Work

In information retrieval (where the proposed method is inspired), locality-sensitive hashing (LSH) is well studied and explored. The aim of LSH is to preserve the similarity between inputs after hashing. This aim is well aligned with that of embedding compression. For example, word similarity can be measured by the cosine distance of their embeddings. If LSH is applied, the hashed emebddings should maintain a similar distance as the original cosine distance but have much lower complexity in the meantime.

A well-known LSH method in image retrieval is ITQ (Gong et al. 2013). However, its application in NLP tasks such as document retrieval is not as successful (Grzegorczyk and Kurdziel 2017). Rather, the authors propose to learn binary paragraph embeddings via a bag-of-words-like model, which essentially computes a binary hash function for the real-valued embedding vectors.

On the other hand, (Shu and Nakayama 2017) propose a compact structure for embeddings by using the gumble softmax. In this approach, each word vector is represented as the summation of a set of real-valued embeddings. This idea amounts to learning a low-rank representation of the embedding matrix.

Pre-trained embeddings may be directly used in deep neural networks (DNN) or serve as initialization (Kim 2014). There exist several compression techniques for DNNs, including pruning (Han, Mao, and Dally 2015) and low-rank compression (Sainath et al. 2013). Most of these techniques requires retraining for specific tasks, thus challenges exist when applying them to unsupervised word embeddings (e.g., GloVe).

(See, Luong, and Manning 2016) successfully apply DNN compression techniques to unsupervised embeddings. The authors use pruning to sparsify embedding vectors, which however requires retraining after each pruning iteration. Although retraining is common when compressing DNNs, it often takes a long time to recover the model performance. Similarly, (Acharya et al. 2019) uses low rank approximation to compress word embeddings, but they also face the same problem to fine-tune a supervised model.

3 Preliminaries

3.1 Iterative Quantization

In this section, we briefly revisit the iterative quantization method by breaking it down into two steps. The first step is to maximize bit variance when transforming given vectors into binary representation. The second step is about minimizing the quantization loss while maintaining the maximum bit variance.

Maximize Bit Variance. Let $\mathbf{X} \in \mathbb{R}^{n \times d}$ be the embedding dictionary, where each row $\mathbf{x}_i^T \in \mathbb{R}^d$ denotes the embedding vector for the i-th word in the dictionary. Assuming that vectors are zero centered ($\sum_{i=1}^{n} \mathbf{x}_i = \mathbf{0}$), ITQ encodes vectors with a binary representation $\{-1, +1\}$ through maximizing the bit variance, which is achieved by solving the following optimization problem:

$$\begin{aligned} \max_{\mathbf{W}} \ & F(\mathbf{W}) = \frac{1}{n}\text{tr}(\mathbf{W}^T\mathbf{X}^T\mathbf{X}\mathbf{W}), \\ \text{s.t. } & \mathbf{W}^T\mathbf{W} = \mathbf{I} \text{ and } \mathbf{B} = \text{sgn}(\mathbf{X}\mathbf{W}), \end{aligned} \tag{1}$$

where $\mathbf{W} \in \mathbb{R}^{d \times c}$ and $c \leq d$ is the dimension of the encoded vectors. Here, $\mathbf{B}$ is the final binary representation of $\mathbf{X}$ and $\text{tr}(\cdot)$ and $\text{sgn}(\cdot)$ are the trace and the sign function, respectively. The problem is the same as that of Principal Component Analysis (PCA) and could be solved by selecting the top c right singular vectors of $\mathbf{X}$ as $\mathbf{W}$.

Minimize Quantization Loss. Given a solution $\mathbf{W}$ to Equation (1), $\mathbf{U} = \mathbf{W}\mathbf{R}$ is also a solution for any orthogonal matrix $\mathbf{R} \in \mathbb{R}^{c \times c}$. Thus, we could minimize the quantization loss via adjusting the matrix $\mathbf{R}$ while maintaining the solution to (1). The quantization loss is defined as the difference between the vectors before and after the quantization:

$$Q(\mathbf{B}, \mathbf{R}) = ||\mathbf{B} - \mathbf{X}\mathbf{W}\mathbf{R}||_F^2, \tag{2}$$

where $||\cdot||_F$ is the Frobenius norm. Note that $\mathbf{B}$ must be binary. The proposed solution in ITQ is an iterative procedure that updates $\mathbf{B}$ and $\mathbf{R}$ in an alternating fashion until convergence. In practice, ITQ turns out able to achieve good performance with early stopping (Gong et al. 2013).

3.2 Isotropy of Word Embedding

In (Arora et al. 2016), isotropy is used to explain the success of PMI based word embedding algorithm, for example GloVe embedding. However, (Mu, Bhat, and Viswanath 2017) find that existing word embeddings are not nearly isotropic but could be improved. The proposed solution is to project word embeddings toward the weak directions rather than the dominant directions, which seems counter-intuitive but in practice works well. The isotropy of word embedding $\mathbf{X}$ is defined as:

$$I(\mathbf{X}) = \frac{\min_{||\mathbf{e}||=1} Z(\mathbf{e})}{\max_{||\mathbf{e}||=1} Z(\mathbf{e})}, \tag{3}$$

where $Z(\cdot)$ is the partition function

$$Z(\mathbf{e}) = \sum_{\mathbf{x}_i \in \mathbf{X}} \exp(\mathbf{e}^T\mathbf{x}_i). \tag{4}$$

The value of $I(\mathbf{X}) \in [0, 1]$ is a measure of isotropy of the given embedding $\mathbf{X}$. A higher $I(\cdot)$ means more isotropic and a better quality of the embedding. It is found making the singular values close to each other can effectively improve embedding isotropy.

4 Proposed Method

The preceding section hints that maximizing the isotropy and maximizing the bit variance are opposite in action: The former intends to make the singular values close by removing the largest singular values, whereas the latter removes the smallest singular values and maintains the largest. Given the success of isotropy in NLP applications, we propose to minimize the quantization loss while improving the isotropy, rather than maximizing the bit variance. We call the proposed method *isotropic iterative quantization*, IIQ.

The key idea of ITQ is based on the observation that $\mathbf{U} = \mathbf{WR}$ is still a solution to the objective function of (1). In our approach IIQ, we show that the orthogonal transformation maintains the isotropy of the input embedding, so that we could apply a similar alternating procedure as in ITQ to minimize the quantization loss. As a result, our method is composed of three steps: maximizing isotropy, reducing dimension, and minimizing quantization loss.

Maximize Isotropy. The isotropy measure $I(\mathbf{X})$ can be approximated as following (Mu, Bhat, and Viswanath 2017) :

$$\hat{I}(\mathbf{X}) = \frac{|\mathbf{X}| - ||\mathbf{1}^T\mathbf{X}|| + \frac{1}{2}\sigma_{\min}^2}{|\mathbf{X}| + ||\mathbf{1}^T\mathbf{X}|| + \frac{1}{2}\sigma_{\max}^2}, \tag{5}$$

where $\sigma_{\min}$ and $\sigma_{\max}$ are the smallest and largest singular values of $\mathbf{X}$, respectively. For $\hat{I}(\mathbf{X})$ to be 1, the middle term $||\mathbf{1}^T\mathbf{X}||$ on both the numerator and the denominator must be zero and additionally $\sigma_{\min} = \sigma_{\max}$. The former requirement can be easily satisfied by the zero-centering given embeddings:

$$\begin{aligned} \mathbf{u} &= \frac{1}{n}\mathbf{1}^T \cdot \mathbf{X} \\ \bar{\mathbf{X}} &= \mathbf{X} - \mathbf{1} \cdot \mathbf{u}^T, \end{aligned} \tag{6}$$

where $||\mathbf{1}^T\bar{\mathbf{X}}|| = 0$. The latter may be approximately achieved by removing the large singular values such that the rest of the singular values are close to each other. A reason why removing the large singular values makes the rest close, is that often the large singular values have substantial gaps while the rest are clustered. However, removing singular components does not change its dimension. We denote the maximized result as $\hat{\mathbf{X}}$.

Dimension Reduction. To make our method more flexible, we perform a dimension reduction afterward by using PCA. This step essentially removes the smallest singular values so that the clustering of the singular values may be further tightened. Note that PCA won't affect the maximized isotropy of given embeddings, since it only works on the singular values that are already closed to each other after previous step. One can treat the dimension as a hyperparameter, tailored for each data set.

Minimize Quantization Loss. Given a solution $\hat{\mathbf{X}}$ to the maximization of (5), we prove that multiplying $\hat{\mathbf{X}}$ with an orthogonal matrix $\mathbf{R}$ results in the same $\hat{I}(\mathbf{X})$. In other words, we could minimize the quantization loss (2) while maintaining the isotropy.

Proposition 1. *If $\hat{\mathbf{X}} \in \mathbb{R}^{n\times d}$ is isotropic and $\mathbf{R} \in \mathbb{R}^{d\times d}$ is orthogonal, then $\mathbf{U} = \hat{\mathbf{X}}\mathbf{R}$ admits $\hat{I}(\mathbf{U}) = \hat{I}(\hat{\mathbf{X}})$.*

Proof. Given that $\mathbf{R}$ is orthogonal, we first prove that $\mathbf{U}$ has the same singular values as does $\hat{\mathbf{X}}$. Let $\hat{\mathbf{X}}$ have the singular value decomposition (SVD)

$$\hat{\mathbf{X}} = \mathbf{P}\text{diag}(\sigma_{\max}, \ldots, \sigma_{\min})\mathbf{Q}, \tag{7}$$

where $\mathbf{P} \in \mathbb{R}^{n\times d}$ and orthogonal matrix $\mathbf{Q} \in \mathbb{R}^{d\times d}$. Let $\mathbf{Q}' = \mathbf{QR}$. Then, we have

$$\mathbf{U} = \mathbf{P}\text{diag}(\sigma_{\max}, \ldots, \sigma_{\min})\mathbf{Q}'. \tag{8}$$

Since $\mathbf{Q}'$ is also orthogonal, Equation (8) gives the SVD of $\mathbf{U}$. Therefore, $\mathbf{U}$ has the same singular values as does $\hat{\mathbf{X}}$.

Moreover, $||\mathbf{1}^T\mathbf{U}|| = ||\mathbf{1}^T\hat{\mathbf{X}}\mathbf{R}|| = 0$, thus $\mathbf{U}$ is also zero-centered. By Equation (5), we conclude $\hat{I}(\mathbf{U}) = \hat{I}(\hat{\mathbf{X}})$. □

With the given proof, we can always use an orthogonal matrix $\mathbf{R}$ to reduce the quantization loss. The iterative optimization strategy as in ITQ (Gong et al. 2013) is adopted to minimize the quantization loss. Two alternating steps lead to a local minimum. First, compute $\mathbf{B}$ given $\mathbf{R}$:

$$\mathbf{B} = \text{sgn}(\hat{\mathbf{X}} \cdot \mathbf{R}). \tag{9}$$

Second, update $\mathbf{R}$ given $\mathbf{B}$. The update minimizes the quantization loss, which essentially solves the orthogonal Procrustes problem. The solution is given by

$$\begin{aligned} \mathbf{S} \cdot \mathbf{\Omega} \cdot \hat{\mathbf{S}}^T &= \text{SVD}(\mathbf{B}^T \cdot \hat{\mathbf{X}} \cdot \mathbf{R}) \\ \mathbf{R} &= \hat{\mathbf{S}} \cdot \mathbf{S}^T, \end{aligned} \tag{10}$$

where SVD$(\cdot)$ is the singular value decomposition function and $\mathbf{\Omega}$ is the diagonal matrix of singular values.

This iterative updating strategy runs until a local optimal solution is found. Fig. 1 shows an example of the quantization loss curve. This result is similar to the behavior of ITQ, the authors of which proposed using early stopping to terminate iteration in practice. We follow the guidance and run only 50 iterations in our experiments.

Overall Algorithm. Our method is an unsupervised approach, which does not require any label supervision. Therefore, it can be applied independently of downstream tasks and no fine tuning is needed. This advantage benefits many problems where embeddings often slow down the learning process because of the high space and computation complexity.

We present the pseudocode of the proposed IIQ method in Algorithm 1. The input D denotes the number of top singular values to be removed, T denotes the number of iterations for minimizing the quantization loss, and O denotes the dimension of the output binary vectors.

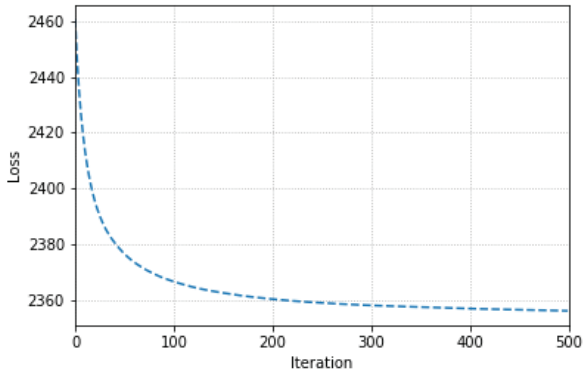


Figure 1: Quantization loss curve of 50000 embedding vectors from a pre-trained CNN model.

The first two lines make zero-centered embedding. Lines 3 to 5 maximize the isotropy. Lines 6 to 8 reduce the embedding dimension, if necessary. Lines 9 to 15 minimize the quantization loss. Within the iteration loop, lines 11 to 12 update $\mathbf{B}$ based on the most recent $\mathbf{R}$, whereas lines 13 to 14 update $\mathbf{R}$ given the updated $\mathbf{B}$. The last line uses the final transformation $\mathbf{R}$ to return the binary embeddings as output.

Algorithm 1: Isotropic Iterative Quantization

Input: $\mathbf{X} \in \mathbb{R}^{n \times d}, D, T, O$
Output: $\mathbf{B}$

1 $\mathbf{u} \leftarrow \frac{1}{n}\mathbf{1}^T \cdot \mathbf{X}$;
2 $\bar{\mathbf{X}} \leftarrow \mathbf{X} - \mathbf{1} \cdot \mathbf{u}^T$;
3 $\mathbf{S} \cdot \mathbf{\Omega} \cdot \hat{\mathbf{S}}^T \leftarrow \text{SVD}(\bar{\mathbf{X}})$;
4 Set top D singular values in $\mathbf{\Omega}$ as 0;
5 $\hat{\mathbf{X}} \leftarrow \mathbf{S} \cdot \mathbf{\Omega} \cdot \hat{\mathbf{S}}^T$;
6 **if** $O < d$ **then**
7 $\quad \hat{\mathbf{X}} \leftarrow \text{PCA}(\hat{\mathbf{X}}, O)$
8 **end**
9 Randomly initialize an orthogonal matrix $\mathbf{R}$;
10 **for** $i \leftarrow 1$ *to* T **do**
11 $\quad \mathbf{U} \leftarrow \hat{\mathbf{X}} \cdot \mathbf{R}$;
12 $\quad \mathbf{B} \leftarrow \text{sgn}(\mathbf{U})$;
13 $\quad \mathbf{S} \cdot \mathbf{\Omega} \cdot \hat{\mathbf{S}}^T \leftarrow \text{SVD}(\mathbf{B}^T \cdot \mathbf{U})$;
14 $\quad \mathbf{R} \leftarrow \hat{\mathbf{S}} \cdot \mathbf{S}^T$;
15 **end**
16 **return** $sgn(\hat{\mathbf{X}} \cdot \mathbf{R})$;

5 Experimental Results

We run the proposed method on pre-trained embedding vectors and evaluate the compressed embedding in various NLP tasks. For some tasks, the evaluation is directly conducted over the embedding (e.g., measuring the cosine similarity between word vectors); whereas for others, a classifier is trained with the embedding. We conduct all experiments in Python by using Numpy and Keras. The environment is Ubuntu 16.04 with Intel(R) Xeon(R) CPU E5-2698.

Table 1: Experiment Configurations.

	Method	Dimension	Comp. Ratio
GloVe	Baseline	1917494×300	1
	Prune	1917494×300	20
	DCCL	$M = 32, K = 256$	32
	NLB	1917494×300	32
	ITQ	1917494×300	32
	IIQ-32	1917494×300	32
	IIQ-64	1917494×150	64
	IIQ-128	1917494×75	128
HDC	Baseline	388723×300	1
	Prune	388723×300	20
	DCCL	$M = 32, K = 128$	29
	NLB	388723×300	32
	ITQ	388723×300	32
	IIQ-32	388723×300	32
	IIQ-64	388723×150	64
	IIQ-128	388723×75	128

Pre-trained Embedding. We perform experiments with the GloVe embedding (Pennington, Socher, and Manning 2014) and the HDC embedding (Sun et al. 2015). The GloVe embedding is trained from 42B tokens of Common Crawl data. The HDC embedding is trained from public Wikipedia. It has a better quality than GloVe because the training process considers both syntagmatic and paradigmatic relations. All embedding vectors are used in the experiment without vocabulary truncation or post-processing.

In addition, we evaluate embedding compression on a CNN model pre-trained with the IMDB data set. Different from the prior case, the embedding from CNN is trained with supervised class labels. We compress the embedding and retrain the model to evaluate performance. This way enables us to compare with other compression methods fairly.

Configuration. We compare IIQ with the traditional ITQ method (Gong et al. 2013), the pruning method (See, Luong, and Manning 2016), deep compositional code learning (DCCL) (Shu and Nakayama 2018) and a recent method (Tissier, Gravier, and Habrard 2019) we name as NLB. The pruning method is set to prune 95% of the words for a similar compression ratio. The DCCL method is similarly configured. We run NLB with its default setting. We train the DCCL method for 200 epochs and set the batch size to be 1024 for GloVe and 64 for HDC. For our method, we set the iteration number T to be 50 since early stopping works sufficiently well. We set the same iteration number for ITQ. We also set the parameter D to be 2 for HDC, and 14 for Glove embedding. Note that we perform all vector operations in real domain on the platform (Jastrzebski, Leśniak, and Czarnecki 2015) and (Conneau and Kiela 2018).

Table 1 lists the experiment configurations with method name, dimension, embedding value type, and compression ratio. The baseline means the original embedding. Our

Table 2: Word Similarity Results.

	Method	MEN	MTurk	RG65	RW	SimLex999	TR9856	WS353
GloVe	Baseline	73.62	64.50	81.71	37.43	37.38	9.67	69.07
	Prune	17.97	22.09	39.66	12.45	-0.37	8.31	14.52
	DCCL	54.46	50.46	63.89	28.04	25.48	7.91	54.55
	NLB	73.99	**64.98**	72.07	40.86	40.52	**14.00**	66.09
	ITQ	57.37	52.93	72.08	25.10	26.23	8.98	55.00
	IIQ-32	**76.43**	63.33	**78.16**	**41.35**	**41.87**	9.80	**72.22**
	IIQ-64	71.55	58.37	74.94	37.61	38.80	12.81	67.99
	IIQ-128	59.25	50.42	62.39	28.71	33.25	12.31	53.56
HDC	Baseline	76.03	65.77	80.58	46.34	40.68	20.71	76.81
	Prune	46.83	41.49	56.14	29.84	26.27	15.27	52.06
	DCCL	68.82	55.78	72.23	**39.33**	35.02	**18.41**	66.09
	NLB	72.06	61.57	72.58	35.45	38.50	11.71	67.20
	ITQ	72.31	61.68	74.70	37.01	37.40	9.69	72.32
	IIQ-32	**74.37**	**66.71**	**78.04**	38.75	**39.35**	9.63	**75.32**
	IIQ-64	66.32	56.73	65.77	35.63	36.22	11.33	72.70
	IIQ-128	55.83	51.33	45.76	32.03	29.45	12.61	58.54

method starts with "IIQ," followed by the compression ratio. The "dimension" column gives the number of vectors and the vector dimension. For DCCL, we list the parameters M and K that determine the compression ratio. Note that we use single precision for real values. The last column shows the compression ratio, which is the the size of the original embedding over that of the compressed one. Thus, the compression from real value to binary is 32. Moreover, we also apply dimension reduction in IIQ so that higher compression ratio is possible.

5.1 Word Similarity

The task measures Spearman's rank correlation between word vector similarity and human rated similarity. A higher correlation means a better quality of the word embedding. The similarity between two words is computed as the cosine of the corresponding vectors, i.e., $\cos(\mathbf{x}, \mathbf{y}) = \mathbf{x}^T\mathbf{y}/(||\mathbf{x}|| \cdot ||\mathbf{y}||)$, where $\mathbf{x}$ and $\mathbf{y}$ are two word vectors. Out-of-vocabulary (OOV) words are replaced by the mean vector.

In this experiment, seven data sets are used, including MEN (Bruni, Tran, and Baroni 2014) with 3000 pairs of words obtained from Amazon crowdsourcing; MTurk (Radinsky et al. 2011) with 287 pairs, focusing on word semantic relatedness; RG65 (Rubenstein and Goodenough 1965) with 65 pairs, an early published dataset; RW (Luong, Socher, and Manning 2013) with 2034 pairs of rare words selected based on frequencies; SimLex999 (Hill, Reichart, and Korhonen 2015) with 999 pairs, aimed at genuine similarity estimation; TR9856 (Levy et al. 2015) with 9856 pairs, containing many acronyms and name entities; and WS353 (Agirre et al. 2009) with 353 pairs of mostly verbs and nouns. The experiment is conducted on the platform (Jastrzebski, Leśniak, and Czarnecki 2015).

Table 2 summarizes the results. The performance of IIQ degrades as the compression ratio increases. This is expected, since a higher compression ratio leads to more loss of information. In addition, our IIQ method consistently achieves better results than ITQ, DCCL, NLB and the pruning method. Particularly, one sees that on the Men data set, IIQ even outperforms the baseline embedding Glove. Another observation is that on TR9856, a higher compression ratio surprisingly yields better results for IIQ. We speculate that the cause is the multi-word term relations unique to TR9856. Interestingly, the pruning method results in negative correlation in SimLex999 for the GloVe embedding. This means that pruning too many small values inside word embedding can drastically destroy the embedding quality.

5.2 Categorization

The task is to cluster words into different categories. The performance is measured by purity, which is defined as the fraction of correctly classified words. We run the experiment using agglomerative clustering and k-means clustering, and select the highest purity as the final result for each embedding. This experiment is conducted on the platform (Jastrzebski, Leśniak, and Czarnecki 2015) where OOV words are replaced by the mean vector.

Four data sets are used in this experiment: Almuhareb-Poesio (AP) (Almuhareb and Poesio 2005) with 402 words in 21 categories; BLESS (Baroni and Lenci 2011) with 200 nouns (animate or inanimate) in 17 categories; Battig (Battig and Montague 1969) with 5231 words in 56 taxonomic categories; and ESSLI2008 Workshop (M Baroni and Lenci 2008) with 45 verbs in 9 semantic categories.

Table 3 lists evaluation results for GloVe and HDC embeddings. One sees that the proposed IIQ method works better than ITQ, DCCL, and the pruning method on all data sets. But NLB sometimes achieves the best result for example on Battig. For ESSLI, IIQ even outperforms the original GloVe and HDC embedding.

Table 3: Categorization Results.

	Method	AP	BLESS	Battig	ESSLI
GloVe	Baseline	62.94	78.50	45.13	57.78
	Prune	38.56	46.00	23.42	42.22
	DCCL	52.24	75.00	36.09	48.89
	NLB	59.45	78.50	**43.39**	**66.67**
	ITQ	58.71	76.50	40.76	48.89
	IIQ-32	**64.18**	**80.00**	41.98	60.00
	IIQ-64	56.22	76.50	37.49	51.11
	IIQ-128	45.02	69.00	31.43	44.44
HDC	Baseline	65.42	81.50	43.18	60.00
	Prune	34.33	48.00	23.28	51.11
	DCCL	55.97	74.50	40.16	53.33
	NLB	59.20	75.50	**41.88**	62.22
	ITQ	57.21	77.50	41.04	55.56
	IIQ-32	**61.69**	**78.00**	41.29	**62.22**
	IIQ-64	48.51	72.50	35.90	53.33
	IIQ-128	43.03	57.50	28.50	62.22

Table 4: Topic Classification Results.

	Method	CR	MPQA	MR	SUBJ
GloVe	Baseline	78.78	87.16	76.42	91.29
	Prune	73.48	81.93	71.97	87.19
	DCCL	77.27	**85.6**	74.74	89.56
	NLB	75.36	85.77	73.01	**89.92**
	ITQ	71.79	84.11	73.18	89.55
	IIQ-32	**77.7**	85.15	**74.96**	89.87
	IIQ-64	75.07	83.02	73.17	88.14
	IIQ-128	72.56	80.55	69.93	84.29
HDC	Baseline	76.40	86.61	75.71	90.86
	Prune	70.97	78.84	67.56	83.58
	DCCL	74.68	84.2	73.32	89.43
	NLB	70.89	84.51	73.18	89.48
	ITQ	73.57	84.44	72.3	89.46
	IIQ-32	**76.32**	**84.77**	**73.51**	**89.91**
	IIQ-64	72.18	82.07	70.32	87.41
	IIQ-128	70.83	77.62	67.89	84.62

5.3 Topic Classification

In this experiment, we perform topic classification by using sentence embedding. The embedding is computed as the average of the corresponding word vectors. The average of binary embedding is fed to the classifier in single precision. Missing words are treated as zero and so are OOV words. In this task, we train a Multi-Layer Perceptron (MLP) as the classifier for each method. Due to the different size of embeddings, we train 10 epochs for all Glove embeddings and 4 epochs for all HDC embedding. Five-fold cross validation is used to report classification accuracy.

Four data sets are selected from (Wang and Manning 2012), including movie review (MR), customer review (CR), opinion-polarity (MPQA), and subjectivity (SUBJ). Similar performance is achieved by using the original embedding. The experiment is conducted on the platform of (Conneau and Kiela 2018).

Table 4 shows the results for each method. Similar to the previous tasks, the proposed IIQ method consistently performs better than ITQ, pruning, and DCCL. The only exception is that for MPQA and SUBJ, DCCL and NLB achieves the best result for the GloVe embedding respectively. As the compression ratio increases, IIQ encounters performance degrade.

5.4 Sentiment Analysis

In this experiment, we evaluate over the embedding input to a pre-trained Convolutional Neural Network (CNN) model on the IMDB data set (Maas et al. 2011). The CNN model follows the Keras tutorial (Chollet and others). We train 50,000 embedding vectors in 300 dimensions. The model is composed of an embedding layer, followed by a dropout layer with probability 0.2, a 1D convolution layer with 250 filters and kernel size 3, a 1D max pooling layer, a fully connected layer with hidden dimension 250, a dropout layer with probability 0.2, a ReLU activation layer, and a single output fully connected layer with sigmoid activation. Moreover, we use adam optimizer with learning rate 0.0001, sentence length 400, batch size 128, and train for 20 epochs. Input embedding fed into CNN is kept fixed (not trainable).

The data set contains 25,000 movie reviews for training and another 25,000 for testing. We randomly separate 5,000 reviews from the training set as validation data. The model with the best performance on the validation set is kept as the final model for measuring test accuracy. Moreover, all results are averaged from 10 runs for each embed-

Table 5: Configurations for IMDB Classification.

Method	Dimension	Comp. Ratio
Baseline	50000×300	1
Prune	50000×300	20
DCCL	$M = 32, K = 32$	27
NLB	50000×300	32
ITQ	50000×300	32
IIQ-32	50000×300	32
IIQ-64	50000×150	64
IIQ-128	50000×75	128

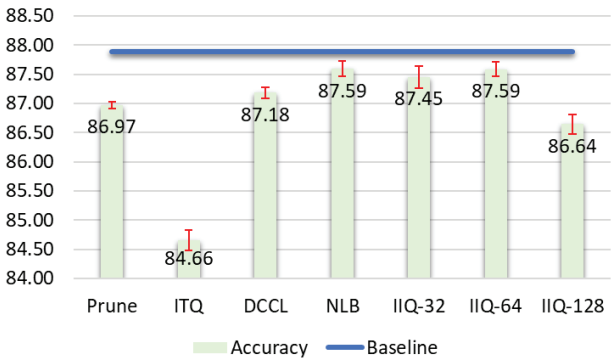
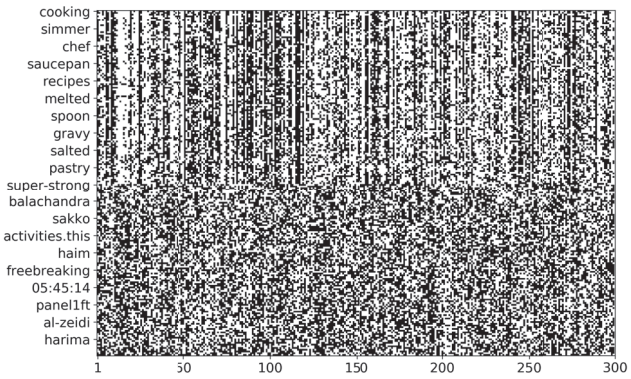
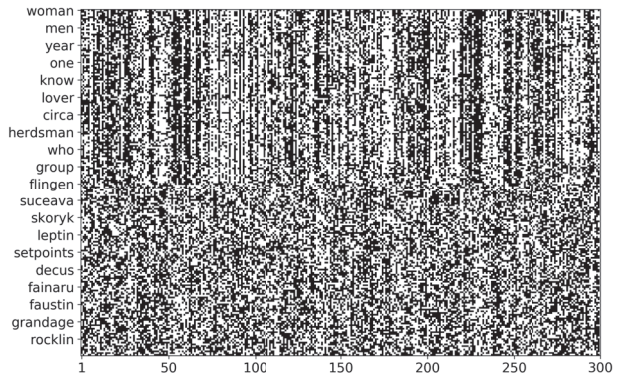


Figure 2: IMDB CNN Test Accuracy Results.



(a) Nearest and furthest 100 words of "cook" in IIQ-GloVe.



(b) Nearest and furthest 100 words of "man" in IIQ-HDC.

Figure 3: Visualizing Binary IIQ Word Embedding.

ding. The baseline model is the pre-trained CNN model with 87.89% accuracy. Table 5 summarizes the configurations for this experiment. All configurations are similar to the previous experiments. The DCCL method is now configured with $M = 32$ and $K = 32$ to achieve a similar compression ratio.

We present in Fig. 2 the result of each embedding. The histogram shows the average accuracy of 10 runs experiments for each method and the error bar shows the standard deviation. One sees that among all compression methods, IIQ achieves the least performance degrade. IIQ with compression ratio 64 is the best.

5.5 Visualization

We visualize the binary IIQ embedding in Fig. 3 The nearest and furthest 100 word vectors are shown. The distance is calculated by the dot product. Fig. 3(a) shows the IIQ-compressed GloVe embedding and Fig. 3(b) shows the IIQ-compressed HDC embedding. The y axis lists every 10 words and the x axis is the dimension of the embedding. One sees that similar word vectors have similar patterns in many dimensions. A white column means that the dimension is zero for all words. A black column means one. Moreover, there is obvious difference between nearest and furthest words.

6 Conclusion

This paper presents an isotropic iterative quantization (IIQ) method for compressing word embeddings. While it is based on the ITQ method in image retrieval, it also maintains the embedding isotropy. We evaluate the proposed method on GloVe and HDC embeddings and show that it is effective for word similarity, categorization, and several other downstream tasks. For pre-trained embeddings that are less isotropic (e.g., GloVe), IIQ performs better than ITQ owing to the improvement on isotropy. These findings are based on a 32-fold (and higher) compression ratio. The results point to promising deployment of trained neural network models with word embeddings on resource constrained platforms in real life.

References

Acharya, A.; Goel, R.; Metallinou, A.; and Dhillon, I. 2019. Online embedding compression for text classification using low rank matrix factorization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 6196–6203.

Agirre, E.; Alfonseca, E.; Hall, K.; Kravalova, J.; Paşca, M.; and Soroa, A. 2009. A study on similarity and relatedness using distributional and wordnet-based approaches. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, 19–27. Association for Computational Linguistics.

Almuhareb, A., and Poesio, M. 2005. Concept learning and categorization from the web. In *proceedings of the annual meeting of the Cognitive Science society*.

Arora, S.; Li, Y.; Liang, Y.; Ma, T.; and Risteski, A. 2016. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics* 4:385–399.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baroni, M., and Lenci, A. 2011. How we blessed distributional semantic evaluation. In *Proceedings of the GEMS 2011 Workshop on GEometrical Models of Natural Language Semantics*, 1–10. Association for Computational Linguistics.

Battig, W. F., and Montague, W. E. 1969. Category norms of verbal items in 56 categories a replication and extension of the connecticut category norms. *Journal of experimental Psychology* 80(3p2):1.

Bruni, E.; Tran, N.-K.; and Baroni, M. 2014. Multimodal distributional semantics. *Journal of Artificial Intelligence Research* 49:1–47.

Chollet, F., et al. Keras documentation, convolution1d for text classification. https://keras.io/examples/imdb_cnn/. Accessed: 2019-08.

Conneau, A., and Kiela, D. 2018. Senteval: An evaluation toolkit for universal sentence representations. *arXiv preprint arXiv:1803.05449*.

dos Santos, C., and Gatti, M. 2014. Deep convolutional neural networks for sentiment analysis of short texts. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, 69–78.

Gong, Y.; Lazebnik, S.; Gordo, A.; and Perronnin, F. 2013. Iterative quantization: A procrustean approach to learning binary codes for large-scale image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35(12):2916–2929.

Grzegorczyk, K., and Kurdziel, M. 2017. Binary paragraph vectors. In *Proceedings of the 2nd Workshop on Representation Learning for NLP*, 121–130.

Han, S.; Mao, H.; and Dally, W. J. 2015. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*.

Hill, F.; Reichart, R.; and Korhonen, A. 2015. Simlex-999: Evaluating semantic models with (genuine) similarity estimation. *Computational Linguistics* 41(4):665–695.

Jastrzebski, S.; Leśniak, D.; and Czarnecki, W. M. 2015. word-embeddings-benchmarks. https://github.com/kudkudak/word-embeddings-benchmarks.

Joulin, A.; Grave, E.; Bojanowski, P.; and Mikolov, T. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Lample, G.; Ballesteros, M.; Subramanian, S.; Kawakami, K.; and Dyer, C. 2016. Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Levy, R.; Ein-Dor, L.; Hummel, S.; Rinott, R.; and Slonim, N. 2015. Tr9856: A multi-word term relatedness benchmark. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, 419–424.

Luong, T.; Socher, R.; and Manning, C. 2013. Better word representations with recursive neural networks for morphology. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, 104–113.

M Baroni, S. E., and Lenci, A. 2008. Bridging the gap between semantic theory and computational simulations: Proceedings of the esslli workshop on distributional lexical semantics. In *Proceedings of the esslli workshop on distributional lexical semantics*.

Maas, A. L.; Daly, R. E.; Pham, P. T.; Huang, D.; Ng, A. Y.; and Potts, C. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. Portland, Oregon, USA: Association for Computational Linguistics.

Mikolov, T.; Chen, K.; Corrado, G.; and Dean, J. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G. S.; and Dean, J. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, 3111–3119.

Mu, J.; Bhat, S.; and Viswanath, P. 2017. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Pennington, J.; Socher, R.; and Manning, C. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 1532–1543.

Radinsky, K.; Agichtein, E.; Gabrilovich, E.; and Markovitch, S. 2011. A word at a time: computing word relatedness using temporal semantic analysis. In *Proceedings of the 20th international conference on World wide web*, 337–346. ACM.

Rubenstein, H., and Goodenough, J. B. 1965. Contextual correlates of synonymy. *Communications of the ACM* 8(10):627–633.

Sainath, T. N.; Kingsbury, B.; Sindhwani, V.; Arisoy, E.; and Ramabhadran, B. 2013. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, 6655–6659. IEEE.

See, A.; Luong, M.-T.; and Manning, C. D. 2016. Compression of neural machine translation models via pruning. *arXiv preprint arXiv:1606.09274*.

Shu, R., and Nakayama, H. 2017. Compressing word embeddings via deep compositional code learning. *arXiv preprint arXiv:1711.01068*.

Shu, R., and Nakayama, H. 2018. Compressing word embeddings via deep compositional code learning. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Sun, F.; Guo, J.; Lan, Y.; Xu, J.; and Cheng, X. 2015. Learning word representations by jointly modeling syntagmatic and paradigmatic relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 136–145.

Tissier, J.; Gravier, C.; and Habrard, A. 2019. Near-lossless binarization of word embeddings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, 7104–7111.

Wang, S., and Manning, C. D. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2*, 90–94. Association for Computational Linguistics.