EmbedTextNet: Dimension Reduction with Weighted Reconstruction and Correlation Losses for Efficient Text Embedding

Dae Yon Hwang[1,2*] **Bilal Taha**[2,3] **Yaroslav Nechaev**[1*]
[1] Amazon Alexa AI [2] University of Toronto [3] Vector Institute
{daeyon.hwang, bilal.taha}@mail.utoronto.ca
{dyhwang, nechaey}@amazon.com

Abstract

The size of embeddings generated by large language models can negatively affect system latency and model size in certain downstream practical applications (e.g. KNN search). In this work, we propose EmbedTextNet, a light add-on network that can be appended to an arbitrary language model to generate a compact embedding without requiring any changes in its architecture or training procedure. Specifically, we use a correlation penalty added to the weighted reconstruction loss that better captures the informative features in the text embeddings, which improves the efficiency of the language models. We evaluated EmbedTextNet on three different downstream tasks: text similarity, language modelling, and text retrieval. Empirical results on diverse benchmark datasets demonstrate the effectiveness and superiority of EmbedTextNet compared to state-of-art methodologies in recent works, especially in extremely low dimensional embedding sizes. The developed code for reproducibility is included in the supplementary material.[1]

1 Introduction

Significant advances in computational resources along with the existence of the huge amount of data rendered large language models (LLM) ubiquitous in various fields, such as Natural Language Understanding (NLU), and vision-language multimodels Du et al. (2022). Even with their relative success in diverse applications, the practical deployment of such models remains challenging with one of the main concerns being the text embedding size (Ling et al., 2016). For example, loading word embedding matrices with 2.5 million tokens and 300 dimensions consumes 6 GB memory on a 64-bit system Raunak et al. (2019). This may breach memory or latency budget in certain highly-constrained practical settings, such as NLU on mobile devices or KNN search in a large index.

Text embeddings, in general, are generated from unlabeled text corpora and applied in several Natural Language Processing (NLP) and information retrieval applications. Embeddings are often precomputed and used in downstream applications. Many post-processing algorithms have been considered over the years to reduce the dimensionality of the resulting vector space. One approach to reduce embedding dimensionality is re-training the language models with the desired dimensionality, this, however, can be costly and reduce model performance. Another approach is to use unsupervised algorithms directly on the pre-computed text embeddings, without requiring any access to labels. Several works exist in the literature including Principal Component Analysis (PCA) Jolliffe and Cadima (2016), Algo Raunak et al. (2019), and Uniform Manifold Approximation and Projection (UMAP) McInnes et al. (2018). While most unsupervised approaches have made noticeable progress, they still suffer from many disadvantages according to our findings. First, they do not perform consistently in different applications. For example, Algo works well, compared to PCA, for Dimensionality Reduction (DR) in a word embedding. However, the roles are switched in the case of a multilingual sentence (m-sentence) embedding. Second, the absolute performance of the baseline methods is significantly lower when extreme reduction is employed (e.g. 16).

In this work, we propose a framework that efficiently reduces the dimensionality of text embedding. We show that the proposed EmbedTextNet, which uses a Variational AutoEncoder (VAE) with modified loss functions, can be used as an add-on module that improves the quality of the text embedding. We also demonstrate that EmbedTextNet can efficiently reduce the size of a text embedding considerably while preserving the quality of the

[*] Work was done outside of Amazon
[1] https://github.com/eoduself/EmbedTextNet

Findings of the Association for Computational Linguistics: ACL 2023, pages 9863–9879
July 9-14, 2023 ©2023 Association for Computational Linguistics

embedding. Our extensive experiments on three downstream applications depict that our method surpasses the original embeddings and state-of-the-art (SOTA).

2 Related Work

Model compression and DR of embedding are receiving more attention nowadays because of their ability to speed the inference process and reduce the model storage space. To compress a model, one can use pruning approaches that aim to repeatedly eliminate the redundant parameters in a model Molchanov et al. (2016). Quantization is another common way that works by limiting the number of bits used to represent the model weights Hubara et al. (2017). Similarly, authors in Li et al. (2017) compressed the input and output embedding layers with a parameter-sharing method. Another approach for model compression is knowledge distillation, where the objective is to make a small model mimic a large pre-trained model by matching the ground-truth and soft labels from the large model Hinton et al. (2015). Recently, researchers considered a block-wise low-rank approximation Chen et al. (2018) which employs statistical features of words to form matrix approximations for embedding and softmax layers. The extension of it was shown in Lee et al. (2021) which is based on word weighting and the k-means clustering methods. Authors in Hrinchuk et al. (2019) considered parameterizing the embedding layers by tensor train decomposition to compress a model with a negligible drop in the result. All the aforementioned methods have the limitation of either requiring re-training the model to accommodate for the changes performed in the network or compromising the accuracy of the learning model. Further, some of these techniques are not studied with pre-trained LLM which limits the scalability.

Unsupervised methods for DR of embedding have gained attention because of their label-free approach that transforms a high-dimensional embedding into a reduced one. The most common starting point is to utilize linear transformation approaches such as PCA or Independent Component Analysis (ICA) Müller et al. (2018); Hyvärinen (2013). These methods aim to project the original high-dimensional embedding into a lower-dimensional one by either maximizing the variance or the mutual information. Another route is to employ nonlinear DR methods including Kernel PCA (KPCA) Schölkopf et al. (1998), t-distributed Stochastic Neighbor Embedding (t-SNE) Van der Maaten and Hinton (2008), and UMAP McInnes et al. (2018). The KPCA employs a kernel function (e.g. Gaussian) to map the original embeddings into a nonlinear space and performs PCA on the projected embeddings. t-SNE performs a nonlinear reduction method to capture the local structure of the high-dimensional embeddings while maintaining a global structure simultaneously. UMAP embeds data points in a nonlinear fuzzy topological representation by neighbor graphs and then, learns a low-dimensional representation while preserving maximum information of this space.

The use of pre-trained LLM is the backbone for several NLP applications such as text retrieval, question-answer, and semantic similarity between pairs. Large pre-trained models such as BERT Devlin et al. (2019), RoBERTa Liu et al. (2019) and XLNet Yang et al. (2019) have shown to provide SOTA performance. However, when we necessitate a smaller embedding size for resource-constrained devices and prompt inference in downstream tasks like text retrieval, it becomes crucial to retrain pre-trained LLMs and fine-tune them to attain the same level of performance as the original model. Using DR, we can circumvent this process while also saving space that would otherwise be reserved for storing retrieval models (i.e. index). One simple approach to reduce the embedding size of these large models is to re-train the model while changing the last hidden layer size. This is inefficient since the re-training process will still, obviously, require enormous resources. A more natural and convenient mechanism is to work directly on the pre-trained large embedding which will eliminate the need for re-training Mu et al. (2017). Recently, the authors in Raunak et al. (2019) combined PCA with a post-processing Mu et al. (2017) to obtain effective DR on word embeddings. However, existing methods fail to work in extreme cases such as a 30-times reduction of the original size.

Therefore, in this work, we propose an algorithm that works directly on pre-computed embeddings to reduce their size even in extreme cases. This is validated on three different applications which are text similarity, language modelling and text retrieval to highlight the effectiveness of the proposed approach in different aspects such as similarity score, inference time and model size.

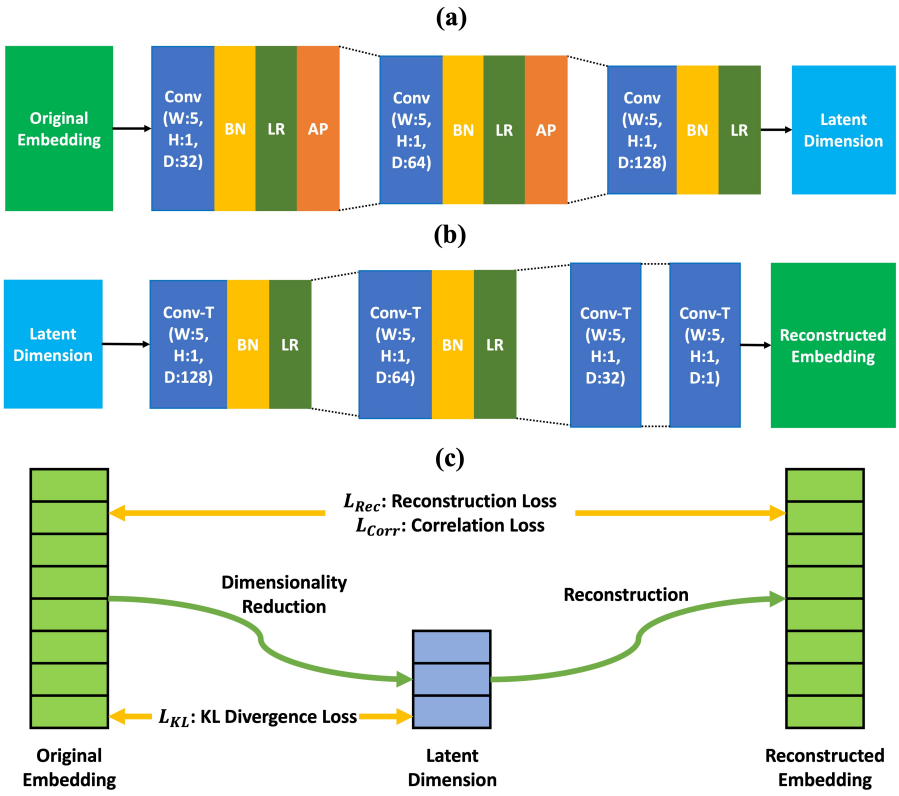

Figure 1: (a) Encoder, (b) Decoder, (c) Loss functions in EmbedTextNet. Conv: Convolutional, Conv-T: Convolutional-transpose, BN: Batch normalization, LR: LeakyReLU, AP: Average Pooling layers respectively.

3 DR for Text Embedding

Ensuring a fast inference (either between devices or user-device interaction) is critical, especially for applications related to NLU such as voice assistants. DR has practical applications in two distinct tasks: (1) Direct application using pre-computed embeddings like GloVe and (2) retrieval task with a built-in index where we use pre-computed embeddings to generate an index for searching. These embeddings can be either contextual or non-contextual, depending on the specific application. To this end, we first introduce EmbedTextNet with modified reconstruction and additional correlation losses. We then show the three most common DR methods as the baselines.

3.1 EmbedTextNet

We propose a VAE model, named EmbedTextNet, with added correlation penalty, L_{Corr} that quantifies the strength of the linear relation between the original embedding and the reconstructed one. The correlation penalty is added to the original losses of the VAE model to enhance the DR of pre-computed embeddings. We mainly consider the latent space of VAE for DR of embeddings. The overall model and loss functions of EmbedTextNet are shown in Figure 1 and detailed in the Appendix.

The input for our EmbedTextNet is the pre-trained embedding. With that in mind, the general loss function of a VAE consists of two parts: Reconstruction and regularizer losses. The first term, L_{Rec}, measures the distance between the original and reconstructed pre-computed embedding. The second term is the Kullback-Leibler (KL) divergence loss, L_{KL}, which acts as a regularizer normalizing and smoothing the latent space to follow a prior distribution (i.e. Gaussian distribution). Originally, KL divergence drives the VAE as a generative model but there is also merit to using it for DR according to the ablation study in Table 7. To balance the two terms, the work in Higgins et al. (2016) proposed a β-VAE which places greater emphasis on the L_{KL} in order to enforce a comparable latent distribution to the original distribution. This weighting value contributes to the production of high-quality synthetic data, indicating that β-VAE is primarily intended for data generation purposes. However, β-VAE is found to be less effective on applications related to DR for text embeddings where significant performance degradation is expected

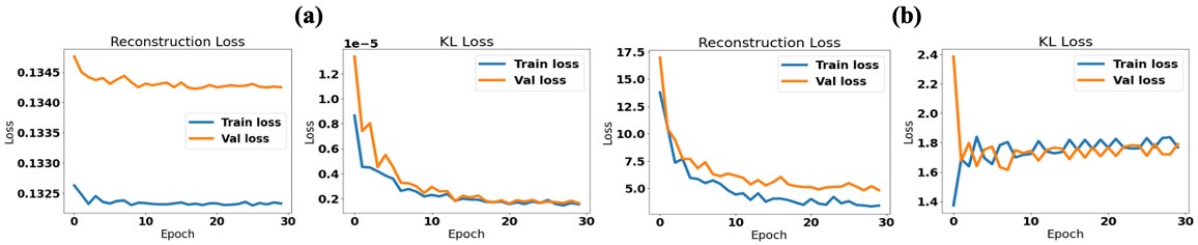

Figure 2: Loss functions before (a) and after (b) weighted in Mean Square Error (MSE)-based reconstruction loss. The graphs are based on DR on word embedding (i.e. GloVe-300D $\rightarrow$ 50D). The increase of epochs does not affect the convergence of losses.

(seen in Table 8). Instead, we place more weight on the reconstruction loss to highlight the similarity between the reconstructed data and the original data. This new weighting value ensures that the latent space contains the critical features of the original data, ensuring that EmbedTextNet is beneficial for dimensionality reduction.

In Figure 2, we show the L_{Rec} and L_{KL} of the reduced word embedding while training VAE. The original VAE losses in (a) are small, especially for the KL loss which almost disappears and it causes the reconstruction term to converge. This makes the training of EmbedTextNet insufficient, resulting in a faulty reconstruction. In addition, the VAE model is not trained properly with the original reconstruction loss since it is almost invariant during the training process. However, weighted reconstruction loss in (b) indicates adequate amounts of loss with convergence at the end. This allows the model to effectively find smaller embeddings by enforcing a large penalty for incorrect decisions. From the empirical result on text similarity in Table 7, we observe the improvement after weighting the reconstruction loss. More experiments are found in the Appendix.

The reconstruction loss is the proximity of two embeddings to gauge their similarity. In the NLP area, we also utilized directional metrics (e.g. cosine similarity) to gauge the similarity between embeddings. Our conception stemmed from the desire to incorporate the additional directional information between the original and reconstructed data to devise a more effective DR approach. To do this, we additionally consider the Pearson correlation coefficient, L_{Corr}, as an added penalty to the VAE losses to measure the strengths and directions of the linear relationship between original and reconstructed embeddings. Specifically, it enables the model to capitalize further on the train set when the reconstruction and KL losses have converged leading to better overall reconstruction performance. From empirical results in Tables 8 and 9, we observe the improvement after including the correlation loss (i.e. EmbedTextNet), compared to other measurements (e.g. cosine-similarity).

Algorithm 1 EmbedTextNet Algorithm

Data: Embedding from train set X_{train} and test set X_{test}, Decreased dimension M, Correlation loss threshold N, Max epochs K

Result: Decreased embedding for train set Y_{train} and test set Y_{test}

Parameters: Training epoch E, Reconstructed train set $X_{re-train}$ and test set $X_{re-test}$, Loss functions of EmbedTextNet L_{All}, Reconstruction L_{Rec}, KL divergence L_{KL}, Correlation L_{Corr}, Weight for reconstruction loss W, Tuning value γ

1. Initialize $Encoder$ and $Decoder$ in EmbedTextNet

2. Train $Encoder$ and $Decoder$:

while $E <= K$ **do**
  $Y_{train} = Encoder(X_{train})$
  $X_{re-train} = Decoder(Y_{train})$
  $Measure\ Loss\ with\ Y_{train},\ X_{re-train}$
  if E == *1* **then**
    $W = \frac{L_{Rec}}{L_{KL}} \cdot \gamma$
    $\gamma = 10\ if\ L_{Rec} < 0.1\ else\ 0.3$
  else
    if $E < N$ **then**
      $L_{All} = W \cdot L_{Rec} + L_{KL}$
    else
      $L_{All} = W \cdot L_{Rec} + L_{KL} + L_{Corr}$
    end
  end
end

3. After training $Encoder$ and $Decoder$:
$Y_{test} = Encoder(X_{test})$
$X_{re-test} = Decoder(Y_{test})$

A detailed description of the EmbedTextNet DR procedure is presented in Algorithm 1. We first generate embeddings from train and test sets of language models without using any layer information from them. During training, the latent and reconstructed embeddings of the train set are used to measure the loss functions of EmbedTextNet. In the first epoch, we set the weight value (W) for the reconstruction loss (L_{Rec}) from the L_{Rec} and KL divergence (L_{KL}) with γ. This helps to compensate for the randomness of parameters from the initialization of the encoder and decoder. The value of γ is determined by the first L_{Rec} value, where we provide a high value if it is small and vice versa for others. This value is crucial for making EmbedTextNet useful in all applications, as each one has a different distribution of embeddings. Equation 1 shows the details of each loss in Algorithm 1.

$$\begin{aligned} L_{Rec} &= E_z\left[log\ p_\theta(O \mid z)\right] \\ L_{KL} &= -E_z\left[log\ q(z \mid O) - log\ p_\theta(z)\right] \\ L_{Corr} &= \frac{\sum_{i=1}^{n}(O_i-\bar{O})(R_i-\bar{R})}{\sqrt{\sum_{i=1}^{n}(O_i-\bar{O})^2}\sqrt{\sum_{i=1}^{n}(R_i-\bar{R})^2}} \end{aligned} \quad (1)$$

where θ is the weight value in EmbedTextNet and L_{Rec} objective is to find θ that maximizes the likelihood of original embedding O given latent dimension z. $q(z \mid O)$ is the distribution of latent dimension given original embedding. O_i, R_i are each dimension of original, reconstructed embeddings respectively and $\bar{O}$, $\bar{R}$ are the average values from original, reconstructed embeddings separately.

3.2 Baselines

We compared EmbedTextNet to three conventional and recent dimensionality reduction methods: PCA, Algo, and UMAP. ***PCA*** is a classic technique for reducing vector dimensionality by projecting it onto a subspace with minimal information loss Jolliffe and Cadima (2016). ***Algo*** aims to eliminate the common mean vector and a few top dominating directions from the text embeddings to make them more distinguishable Raunak et al. (2019). ***UMAP*** is a manifold learning approach based on Riemannian geometry and algebraic topology that is similar to t-SNE in visualization quality and preserves more of the global structure while improving runtime McInnes et al. (2018).

4 Results and Discussion

We showcase the DR usefulness of EmbedTextNet on three different applications: Text similarity for measuring the mimicry of reduced embedding with regards to the original one, language modelling for understanding the reproducibility of original data from DR and text retrieval for finding the effectiveness of DR in terms of frugality and performances. We also conduct an ablation study for the hyperparameters and structure of EmbedTextNet. Finally, the computational cost for EmbedTextNet is detailed in Appendix A. In general, during training, EmbedTextNet takes a longer time compared to PCA and Algo, and comparable time to UMAP. Yet, at inference, it has a similar inference time with superior performance. This emphasizes the capabilities and potential of EmbedTextNet for DR.

4.1 Datasets

We evaluated the proposed DR approach on various datasets for each application.

Word Similarity: We consider diverse databases of word pairs with similarity scores rated manually by humans from Faruqui and Dyer (2014). We evaluate similarity between word embeddings using cosine-similarity and measure the correlation between the cosine-similarity scores and human ranking using Spearman's Rank Correlation (SRC, ρ x 100). ***Sentence Similarity***: We use several datasets for similarity evaluation between sentence pairs: the Semantic Textual Similarity (STS) tasks from 2012-2016 Agirre et al. (2012, 2013, 2014, 2015, 2016), the STS Benchmark (STS-B) Cer et al. (2017), and the SICK-Relatedness (SICK-R) datasets Marelli et al. (2014) using SentEval. [2] ***M-Sentence Similarity***: To measure multilingual similarity, we use the recent extension of multilingual STS 2017 database Cer et al. (2017); Reimers and Gurevych (2020). It contains sentence pairs with semantic similarity scores in different languages: AR-AR, EN-EN, and ES-ES for monolingual pairs and AR-EN, ES-EN, EN-DE, EN-TR, FR-EN, IT-EN, and NL-EN for cross-lingual pairs.[3] For both sentence and m-sentence evaluations, we evaluate the SRC between the cosine-similarity of sentence pairs and the ground truth from measured semantic similarity. ***Language Modelling:*** We use WikiText-2 dataset Merity et al. (2016) for language modeling collected from verified good and featured articles on Wikipedia. ***Text Retrieval:*** Lastly, we consider NYTimes Aumüller

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

[3] AR: Arabic, EN: English, ES: Spanish, DE: German, TR: Turkish, FR: French, IT: Italian, NL: Dutch

et al. (2018); NYT, a benchmark dataset for Approximate Nearest Neighbor (ANN) that is a bag of words generated from NYTimes news articles.

4.2 Experimental Setting

In all downstream tasks, we used the training set to train the DR models and the testing set solely to evaluate their generalization performance. We trained each model three times with different random seeds to account for random initialization and averaged the results. For the text similarity task, we mainly used SRC (ρ x 100) to measure the strength and direction of a relationship between two ranked variables. For language modeling, we used perplexity and for text retrieval, we measured the index size, time for searching queries and recall at top-10 (Recall@10). To ensure fairness, we used the same models and datasets for all baselines and DRs. We also report the standard deviation (STD) with each performance measurement as a statistical analysis.

Text Similarity Application: To generate word embeddings, we considered the pre-trained GloVe Pennington et al. (2014) trained on Wikipedia 2014, Gigaword 5, and Twitter corpus. We employed 80% of the dictionary of GloVe for training DR to investigate the generalizability of it for unseen data (i.e. 20% of dictionary). For task evaluation, we utilized the entire dictionary to extract the reduced embeddings for each word to ensure all words were included in the task. To produce sentence embeddings, we employed the pre-trained Sentence-BERT (SBERT) Reimers and Gurevych (2019) trained on Wikipedia and NLI datasets. There are two scenarios for measuring sentence similarity with DR. (1) No dedicated training set is provided in STS 2012 - 2016 Agirre et al. (2012, 2013, 2014, 2015, 2016). Here, we employed a cross-dataset evaluation protocol. For example, we used STS 2013 - 2016 as train set when measuring the result on STS 2012. (2) Dedicated training and testing sets are provided in STS-B Cer et al. (2017) and SICK-R Marelli et al. (2014). In this case, we used the offered train set for training DR models and evaluated them on the provided test set. To generate m-sentence embeddings, we used the pre-trained multilingual Sentence-BERT (mSBERT) Reimers and Gurevych (2020) trained on parallel data for 50+ languages Tiedemann (2012). For train set, we used the offered monolingual pairs of AR-AR, ES-ES and the translated sentences of ES-ES into EN, DE, TR, FR, IT, NL using Google Translator since we do not have monolingual pairs for them. The provided EN-EN dataset was removed from train set since most cross-lingual datasets were generated from translating one sentence of EN-EN Reimers and Gurevych (2020). For test set, all the cross-lingual pairs are considered.

Language Modelling Application: Using the codebase in Pytorch examples,[4] we employed 2-layered LSTM model with *100* hidden units, *50%* dropout after the encoder and used the dedicated train set in WikiText-2 for training a model. We considered the embedding of the train set from the encoder to train the DR approaches and then, implemented them on the embeddings of the test set from the encoder to get the reduced ones. Finally, we did the inverse transform for the reduced embeddings using DR and provided it to the decoder for performance evaluation. Compared to other tasks, we focus on measuring the reproducibility of original data from DR in this task.

Text Retrieval Application: Here, we employed the Hierarchical Navigable Small Worlds (HNSW) Malkov and Yashunin (2016) in Faiss library Johnson et al. (2019). The used hyperparameters in the HNSW index are *12* as the number of neighbors in the graph, *120* as the depth of exploration for building the index, and *16* as the depth of exploration of the search. To build the index, we used a train set in the NYTimes dataset and considered a test set as the queries for searching. Also, train and test sets in this dataset are used for training and testing in DR approaches. To this end, we used the reduced embeddings for building the HNSW index and searching the queries to understand the effectiveness of DR to save resources and performance.

4.3 Word Similarity with DR

For word embeddings, we simulate two different cases of DR on GloVe which are (1) GloVe-300D $\rightarrow$ 50D and (2) GloVe-50D $\rightarrow$ 10D (the left and right sides represent the original and reduced embeddings dimension (D) after DR). The overall performance for EmbedTextNet and the baseline models are found in Table 1. For (1), we noticed in general that PCA and UMAP methods perform badly compared to Algo after reducing the embedding dimension. In most cases, EmbedTextNet outperforms other reduction approaches and shows better results even compared to GloVe-50D (i.e.

[4] https://github.com/pytorch/examples/tree/main/word_language_model

Table 1: Word similarity (ρ x 100 (STD)) with dimensionalty reduction. Here, embedding is decreased from GloVe-300D $\rightarrow$ 50D and GloVe-50D $\rightarrow$ 10D. Bold means the best result among dimensionality reductions.

Embeddings	YP	VERB	SimLex	353-SIM	TR-3K	MTurK-771	353-ALL	MTurk-287	RW	RG-65	MC-30	353-REL	Average
GloVe-300D	56.13	30.51	37.05	66.38	73.75	65.01	60.54	63.32	41.18	76.62	70.26	57.26	58.17 (0)
GloVe-50D	37.72	25.03	26.46	57.32	65.23	55.42	49.93	61.88	34.03	60.21	56.3	46.56	48.01 (0)
PCA-50D	24.34	24.92	16.28	31.36	41.74	30.70	24.91	43.98	17.43	52.97	51.24	23.12	31.92 (0.75)
Algo-50D	37.05	**35.5**	22.75	58.92	60.84	49.49	53.58	54.07	32.94	54.97	56.88	45.24	46.85 (1.64)
UMAP-50D	14.97	21.94	23.88	36.89	35.98	29.34	29.34	35.99	9.95	32.87	17.65	20.67	25.79 (1.7)
EmbedTextNet -50 D	**39.62**	34	**30.58**	**62.64**	**65.92**	**54.27**	**55.6**	**62**	**34.45**	**61**	**62.07**	**53.05**	**51.27 (0.46)**
GloVe-25D	6.54	16.21	7.05	47.95	44.37	38.72	35.44	48.76	27.43	49.63	43.11	29.72	32.91 (0)
PCA-10D	**21.84**	17.22	9.58	24.21	32.13	22.94	16.16	35.61	14.51	38.44	28.09	15.47	23.02 (0.28)
Algo-10D	5.39	16.07	6.95	23.35	26.90	18.71	18.48	22.77	13.04	25.86	24.00	12.57	17.84 (2.13)
UMAP-10D	16.01	21.18	20.04	33.39	39.09	28.94	22.82	35.12	18.79	32.81	35.85	19.15	26.93 (2)
EmbedTextNet -10D	16.47	**26.72**	**20.5**	**48.2**	**51.94**	**38.09**	**38.88**	**54.13**	**24.52**	**49.7**	**44**	**33.01**	**37.18 (0.38)**

Table 2: Comparison between GloVe and EmbedTextNet in terms of resources and performances. Inference means the measurement time for word similarity task and it is done using Intel(R) Xeon(R) CPU @ 2.20 GHz.

Embeddings	Vocabulary Size (MB)	Inference (sec)	Avg. SRC (ρ x 100 (STD))
GloVe-300D	1126	59.9	58.17 (0)
GloVe-200D	729	42.29	55.12 (0)
GloVe-50D	185	15.13	48.01 (0)
EmbedTextNet-100D	367	25.23	56.41 (0.42)
EmbedTextNet-50D	185	15.1	51.27 (0.46)

re-training the model to reduce dimension size). This demonstrates the ability of EmbedTextNet to effectively decrease the dimension of word embedding without the need for re-training where it is more obvious when the reduced dimension is getting smaller. For (2), the performance gap between EmbedTextNet and other DR methods is expanding in most databases. In addition, EmbedTextNet is the only model that reveals superiority compared to GloVe-25D. In this task, we need a vocabulary from pre-trained word embeddings during inference and thus, EmbedTextNet can be useful to reduce the memory and inference time with similar or better performances. This is shown in Table 2, especially when EmbedTextNet-100D outperforms GloVe-200D. Thus, we confirm the usefulness of EmbedTextNet for the word similarity task in terms of resources and performance.

4.4 Sentence Similarity with DR

We tested two cases of dimensionality reductions on sentence embeddings: (1) Reducing SBERT-1024D to 128D and (2) reducing SBERT-1024D to 16D. The overall performance can be found in Table 3. PCA mostly performed better than other methods in both cases, and EmbedTextNet had the best overall performance. This is especially evident in the larger reduction (2). EmbedTextNet also performed better than using the average of word embeddings (i.e. Avg. GloVe-300D, BERT-768D) with lower dimensional embeddings. These results show that EmbedTextNet effectively generates lower dimensional word and sentence embeddings while maintaining good performance.

Table 3: Sentence similarity (ρ x 100 (STD)) with dimensionalty reduction when SBERT-1024D $\rightarrow$ 128D and SBERT-1024D $\rightarrow$ 16D. Bold describes the best result among dimensionality reductions. The evaluations of Avg. GloVe and BERT are done using SentEval.

Embeddings	STS12-16	STS-B	SICK-R
SBERT-1024D	73.32 (0)	75.35 (0)	80.33 (0)
SBERT-768D	70.9 (0)	73.63 (0)	79.27 (0)
PCA-128D	72.53 (0.05)	78.19 (0.33)	76.75 (0.35)
Algo-128D	73.74 (0.06)	77.74 (1.27)	73.63 (0.32)
UMAP-128D	40.33 (0.12)	49.25 (1.24)	46.39 (1.14)
EmbedTextNet-128D	**75.46 (0.05)**	**79.39 (0.28)**	**77.05 (0.47)**
PCA-16D	63.43 (0.09)	72.21 (0.05)	71.29 (0.62)
Algo-16D	60.58 (0.1)	71.4 (1.63)	66.65 (0.18)
UMAP-16D	40.67 (0.88)	49.23 (0.3)	54.41 (0.29)
EmbedTextNet-16D	**66.56 (0.07)**	**73.8 (0.64)**	**72.55 (0.39)**
Avg. GloVe-300D	55.35 (0)	62.96 (0)	71.83 (0)
Avg. BERT-768D	57.24 (0)	64.48 (0)	73.5 (0)

4.5 Multilingual Sentence Similarity with DR

We tested the proposed method in a multilingual setting by considering two use cases: (1) Reducing mSBERT-768D to 64D and (2) reducing mSBERT-768D to 16D. The results can be found in Table 4. In the first case (1), PCA had a small decrease in performance compared to Algo and UMAP, while EmbedTextNet had the best performance overall, except for the case of AR-EN. However, in the extreme reduction case (2), EmbedTextNet had the best performance in all languages. This highlights the effectiveness of EmbedTextNet in reducing the size of embeddings, especially when a large reduction is needed. Therefore, EmbedTextNet can be useful in reducing the dimension of embeddings in

Table 4: M-sentence similarity (ρ x 100 (STD)) with dimensionalty reduction. Here, embedding is decreased from mSBERT-768D $\rightarrow$ 64D and mSBERT-768D $\rightarrow$ 16D. Bold denotes the best result among dimensionality reductions.

Embeddings	AR-EN	ES-EN	EN-DE	EN-TR	FR-EN	IT-EN	NL-EN	Average
mSBERT-768D	80.85	86.11	83.28	74.9	81.17	84.24	82.51	81.87 (0)
mSBERT-384D	81.22	84.44	84.22	76.74	76.59	82.35	81.71	81.04 (0)
PCA-64D	**77.47**	83.22	79.57	71.08	78.65	81.14	79.55	78.67 (0.04)
Algo-64D	69.23	75.47	77.53	63.85	74.95	76.78	76.71	73.5 (0.09)
UMAP-64D	46.37	44.02	51.13	43.2	43.05	47.2	44.76	45.68 (1.23)
EmbedTextNet-64D	77.07	**83.44**	**82.96**	**71.6**	**80.32**	**83.4**	**81.2**	**80 (0.33)**
PCA-16D	71.08	74.85	70.15	67.7	69.55	71.55	69.39	70.61 (0.02)
Algo-16D	57.81	64.59	69.21	56.77	67.46	70.34	71.22	65.54 (0.28)
UMAP-16D	46.31	39.88	47.99	42.18	41.93	46.47	44.24	44.14 (0.42)
EmbedTextNet-16D	**73.55**	**79.56**	**75.67**	**70.71**	**75.82**	**77.42**	**75.36**	**75.44 (0.4)**

multilingual applications.

4.6 Language Modelling with Reconstructed Data from DR

Unlike other model compression works, DR cannot directly reduce the size of model parameters for language modelling since we do not access any parameter or layer information. Instead, we can reduce the dimension of embeddings but can also reconstruct them to the original dimension by inverse transform in DR approaches. In this task, we measure how close the perplexity is when starting from the reduced-dimension vector where the results are covered in Table 5. In all reduced cases, PCA always shows better perplexity than Algo and UMAP approaches while EmbedTextNet outperforms all of them. Therefore, the reconstructed data from EmbedTextNet has a high quality that follows the original data very well.

Table 5: Perplexity in language modelling. UMAP is only considered for the smallest case since authors McInnes et al. (2018) did not recommend for reduced dimension more than 8 because of low performance.

Embeddings	Perplexity (STD)
Original-100D	159.02 (0)
PCA-30D $\rightarrow$ 100D	180.14 (2.02)
Algo-30D $\rightarrow$ 100D	212.53 (2.76)
EmbedTextNet-30D $\rightarrow$ 100D	**163.47 (2.45)**
PCA-10D $\rightarrow$ 100D	267.12 (2.21)
Algo-10D $\rightarrow$ 100D	354.38 (2.39)
EmbedTextNet-10D $\rightarrow$ 100D	**180.81 (2.34)**
PCA-5D $\rightarrow$ 100D	399.51 (2.24)
Algo-5D $\rightarrow$ 100D	476.38 (1.68)
UMAP-5D $\rightarrow$ 100D	1236.84 (6.25)
EmbedTextNet-5D $\rightarrow$ 100D	**208.32 (2.06)**

4.7 Text Retrieval with DR

For text retrieval, we need a space for saving the embeddings (i.e. index size) and a search time for queries which can be saved using DR approaches. In Table 6, we reveal the results in text retrieval task. First of all, we can see that PCA and EmbedTextNet in 64 dimension case enhance the recall values from the original one since they generate compact informative embeddings which help to build a distinguishable index between different keys. It is obvious that we can decrease the index size and search time for each query significantly after DR implementations where EmbedTextNet shows the best results in terms of recall@10. In real-world applications, only an encoder of EmbedTextNet is needed to reduce query dimension and it requires at most 2 MB of space, making it a beneficial approach for text retrieval tasks to achieve efficiency and similar or better performance.

Table 6: Retrieval task inNYTimes benchmark where searching is done using Intel(R) Xeon(R) CPU @ 2.20 GHz. Encoder size of EmbedTextNet is up to 2 MB (0.2M parameter).

Embeddings	Index Size (MB)	Search (msec)	Recall@10 (%) (STD)
Original-256D	315	0.158	54.85 (0)
PCA-64D	102	0.056	55.17 (0.66)
Algo-64D		0.061	54.22 (0.73)
UMAP-64D		**0.021**	34.27 (0.23)
EmbedTextNet-64D		0.045	**57 (0.36)**
PCA-32D	**67**	0.055	48.55 (1.59)
Algo-32D		0.06	45.36 (0.35)
UMAP-32D		**0.017**	34.61 (0.5)
EmbedTextNet-32D		0.035	**50.5 (0.11)**

4.8 Ablation Study

Since EmbedTextNet uses a modified reconstruction loss with a distance-based penalty, we investigated different distance metrics such as Huber, Hinge, Mean Absolute Error (MAE) and MSE, to find and verify the effectiveness of the selected one for text embedding applications. The investigation is shown in Table 16 in the Appendix which confirms that MSE is the most suitable selection for dimensionality reduction cases.

We also tested different values of γ to find the

Table 7: The effect of KL divergence (i.e. VAE) and modified reconstruction loss (i.e. including W in Algorithm 1) when GloVe-300D $\rightarrow$ 150D in ρ x 100.

Type	Word
AE	56.05
AE with modified reconstruction loss	56.86
VAE with modified reconstruction loss	**57.95**

Table 8: Comparison (ρ x 100) between other methods and EmbedTextNet when mSBERT-768D $\rightarrow$ 16D. Correlation loss threshold N is shown in Algorithm 1.

Type	M-sentence
β 20-VAE	3.67
Weighting manually more (x10) on reconstruction loss	74.30
VAE with cosine-similarity	74.06
EmbedTextNet, $N = 1$	75.00
EmbedTextNet, $N = 5$	**75.44**
EmbedTextNet, $N = 15$	74.60

optimal one for each application. The results are detailed in Table 13 - 15 in the Appendix. We assigned larger γ for $L_{Rec} < 0.1$ as it means that the reconstructed output is very similar to the original one. This also indicates that the latent space of EmbedTextNet follows the prior distribution well (i.e. small KL divergence) and a larger γ is needed to prevent convergence without sufficient learning in EmbedTextNet. The best values (i.e. $\gamma = 10 \; if \; L_{Rec} < 0.1 \; else \; 0.3$) were used throughout the evaluation process for different applications.

In Table 7, we compare the use of KL divergence and modified reconstruction loss. As we can see, using the modified reconstruction loss with AE outperforms the naive AE and including the KL divergence (i.e. VAE) shows further improvements. In Table 8, we compare the selected EmbedTextNet with other approaches in a multilingual sentence similarity. We can see that placing more emphasis on KL divergence (i.e. β-VAE) is not suitable for DR applications. Also, giving more emphasis on reconstruction loss and using cosine-similarity (i.e. Tables 8 and 9) are not helpful, unlike the correlation loss (i.e. EmbedTextNet, $N = 5$). This confirms that correlation loss has its own advantage in achieving better DR. Lastly, we confirm that the correlation loss should be included after a few epochs (i.e. $N = 5$) to achieve better performance. More details are included in the Appendix.

Table 9: Comparison (ρ x 100) between cosine similarity and correlation penalty.

Type	Cosine Similarity	Correlation Penalty
GloVe-50D $\rightarrow$ 10D	35.08	**37.18**
SBERT-1024D $\rightarrow$ 16D	65.63	**68.45**
mSBERT-768D $\rightarrow$ 16D	74.06	**75.44**

5 Conclusions

We proposed EmbedTextNet, an efficient add-on network that shows improved retrieval performance compared to previous DR approaches. It is based on a VAE with an improved reconstruction process by using a weighted reconstruction loss with a correlation-based penalty. EmbedTextNet can be applied to any language model without altering its structure, making it suitable for practical applications such as text retrieval. Our evaluations on various applications consistently show superior performance, particularly in cases where extreme reduction is needed.

6 Limitations

There are three predictable limitations in the developed EmbedTextNet. First, while we have performed a thorough evaluation of EmbedTextNet on various downstream tasks, it is still a general-purpose approach and its effectiveness on specific tasks or in specific domains may vary. Thus, further research is needed to fully understand its capabilities and limitations in different contexts.

Second, as mentioned, EmbedTextNet is most suitable for scenarios where the embedding is saved during inference, such as text retrieval or similarity measurement when the fixed embedding is saved with a vocabulary (e.g. GloVe). However, it may not be as effective in scenarios where the embedding needs to be decoded back to its original form, such as text generation.

Third, the effectiveness of EmbedTextNet is evident on a large embedding dimension, and it may decrease when working with a small embedding dimension even if it was still better than other SOTA in our experiments (e.g. GloVe-50D $\rightarrow$ 10D in Table 1). This limitation is due to the fact that EmbedTextNet is based on a VAE architecture, which is known to perform better on high-dimensional data. Therefore, it is better to compare the performances of EmbedTextNet with other SOTA and choose the right one according to the researchers' usage.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Inigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, et al. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *Proceedings of the 9th international workshop on semantic evaluation (SemEval 2015)*, pages 252–263.

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M Cer, Mona T Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2014. Semeval-2014 task 10: Multilingual semantic textual similarity. In *SemEval@ COLING*, pages 81–91.

Eneko Agirre, Carmen Banea, Daniel Cer, Mona Diab, Aitor Gonzalez Agirre, Rada Mihalcea, German Rigau Claramunt, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval-2016. 10th International Workshop on Semantic Evaluation; 2016 Jun 16-17; San Diego, CA. Stroudsburg (PA): ACL; 2016. p. 497-511.* ACL (Association for Computational Linguistics).

Eneko Agirre, Daniel Cer, Mona Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In ** SEM 2012: The First Joint Conference on Lexical and Computational Semantics–Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 385–393.

Eneko Agirre, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. * sem 2013 shared task: Semantic textual similarity. In *Second joint conference on lexical and computational semantics (* SEM), volume 1: proceedings of the Main conference and the shared task: semantic textual similarity*, pages 32–43.

Martin Aumüller, Erik Bernhardsson, and Alexander Faithfull. 2018. Ann-benchmarks: A benchmarking tool for approximate nearest neighbor algorithms.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055*.

Patrick Chen, Si Si, Yang Li, Ciprian Chelba, and Cho-Jui Hsieh. 2018. Groupreduce: Block-wise low-rank approximation for neural language model shrinking. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Yifan Du, Zikang Liu, Junyi Li, and Wayne Xin Zhao. 2022. A survey of vision-language pre-trained models. *arXiv preprint arXiv:2202.10936*.

Manaal Faruqui and Chris Dyer. 2014. Improving vector space word representations using multilingual correlation. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 462–471.

Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework.

Geoffrey Hinton. Neural networks for machine learning.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

Oleksii Hrinchuk, Valentin Khrulkov, Leyla Mirvakhabova, Elena Orlova, and Ivan Oseledets. 2019. Tensorized embedding layers for efficient model compression.

Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. 2017. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898.

Aapo Hyvärinen. 2013. Independent component analysis: recent advances. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 371(1984):20110534.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547.

Ian T Jolliffe and Jorge Cadima. 2016. Principal component analysis: a review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374(2065):20150202.

Jong-Ryul Lee, Yong-Ju Lee, and Yong-Hyuk Moon. 2021. Block-wise word embedding compression revisited: Better weighting and structuring. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4379–4388, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhongliang Li, Raymond Kulhanek, Shaojun Wang, Yunxin Zhao, and Shuang Wu. 2017. Slim embedding layers for recurrent neural language models.

Shaoshi Ling, Yangqiu Song, and Dan Roth. 2016. Word embeddings with limited memory. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 387–392.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Yu Malkov and Dmitry Yashunin. 2016. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PP.

Marco Marelli, Stefano Menini, Marco Baroni, Luisa Bentivogli, Raffaella Bernardi, and Roberto Zamparelli. 2014. A sick cure for the evaluation of compositional distributional semantic models. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 216–223.

Leland McInnes, John Healy, and James Melville. 2018. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models.

Pavlo Molchanov, Stephen Tyree, Tero Karras, Timo Aila, and Jan Kautz. 2016. Pruning convolutional neural networks for resource efficient inference. *arXiv preprint arXiv:1611.06440*.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. 2017. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*.

Klaus-Robert Müller, Sebastian Mika, Koji Tsuda, and Koji Schölkopf. 2018. An introduction to kernel-based learning algorithms. In *Handbook of Neural Network Signal Processing*, pages 4–1. CRC Press.

UCI Machine Learning Repository: Bag of words NYT. Nytimes dataset.

Jeffrey Pennington, Richard Socher, and Christopher D Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

Vikas Raunak, Vivek Gupta, and Florian Metze. 2019. Effective dimensionality reduction for word embeddings. In *Proceedings of the 4th Workshop on Representation Learning for NLP (RepL4NLP-2019)*, pages 235–243.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Nils Reimers and Iryna Gurevych. 2020. Making monolingual sentence embeddings multilingual using knowledge distillation. *arXiv preprint arXiv:2004.09813*.

Bernhard Schölkopf, Alexander Smola, and Klaus-Robert Müller. 1998. Nonlinear component analysis as a kernel eigenvalue problem. *Neural computation*, 10(5):1299–1319.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218. Citeseer.

Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of machine learning research*, 9(11).

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Table 10: EmbedTextNet structure consisting of (a) Encoder and (b) Decoder. BN, LeakyReLu and Average (Avg) Pooling are applied after each operation. Here, word means the text similarity and retrieval with GloVe and Faiss respectively.

(a)

Layer	Operation	Kernel Size (W x H x D)	Stride	BN, LeakyReLu, Avg Pooling
1	Conv #1	5 x 1 x 32	1	Yes
2	Conv #2	5 x 1 x 64		
3	Conv #3	5 x 1 x 128		Yes, except pooling
4	Dense	-	-	No

(b)

Layer	Operation	Kernel Size (W x H x D)	Stride	BN, LeakyReLu
1	Dense	-	-	No
2	Conv-T #1	5 x 1 x 128	word: 1 others: 2	Yes
3	Conv-T #2	5 x 1 x 64		
4	Conv-T #3	5 x 1 x 32		No
5	Conv-T #4	5 x 1 x 1	1	

A Notes on Reproducibility

A.1 Total Computational Budget and Infrastructure used

To train the baselines, we employed the Intel(R) Xeon(R) CPU @ 2.20 GHz and to train the EmbedTextNet, we used the Tesla P100-PCIE for text similarity tasks and Tesla T4 for language modelling and text retrieval tasks. All of them used RAM 25 GB and we trained three times with different seeds for getting the averaged results. For training EmbedTextNet in text similarity applications, it took up to 54 minutes for GloVe, 94 minutes for all cross-validations in STS12-16, 12 minutes for STS-B, 12 minutes for SICK-R, 8 minutes for 8 multilingual languages. For training EmbedTextNet in other applications, it required up to 53 minutes for language modelling and 61 minutes for text retrieval. The time consumption for measuring text similarity in all applications is very small to be negligible while building the language modelling model (i.e. 2 layers of LSTM) and Faiss index took up to 39 minutes and 9 minutes respectively.

A.2 Details of EmbedTextNet Structure

In Table 10, we explain the structure of encoder and decoder in EmbedTextNet. Furthermore, we used RMSprop optimizer (Hinton) with $lr = 2e^{-3}$ while training our models.

A.3 Hyperparameters and Size Estimation in Dimensionality Reductions

In Table 11, we cover all the hyperparameters in dimensionality reductions which are based on the empirical results. The number of parameters used by EmbedTextNet is up to 4M for GloVe, 3M for SBERT, 1M for mSBERT in text similarity task, 7.6M in language modelling and 2.5M in text retrieval.

Table 11: Hyperparameters in dimensionality reductions.

Method	Parameter	Setting
PCA	svd_solver	auto
	iterated_power	auto
	whiten	false
Algo	n_removed_ top_components	7
	svd_solver	auto
	iterated_power	auto
	whiten	false
UMAP	n_neighbors	15
	metric	euclidean
	n_epochs	auto
	learning_rate	1
	min_dist	0.1
Embed TextNet	learning_rate	2e-3
	n_epochs	text similarity: 30 others: 40
	corr_thres (N in Algorithm 1)	5
	batch_size	SBERT: 32, mSBERT: 16 others: 256

Table 12: Details of datasets used. Here, train and test sets are divided according to the usage in this paper.

Datasets	Division	Size	Downstream
Word Similarity	Train	-	Text Similarity
	Test	8K	
STS 2012 - 2016	Train	-	
	Test	26K	
STS-B	Train	5K	
	Test	1K	
SICK-R	Train	4K	
	Test	4K	
Extension of Multilingual STS 2017	Train	2K	
	Test	1K	
WikiText-2	Train	2M	Language Modelling
	Test	245K	
NYTimes	Train	290K	Text Retrieval
	Test	10K	

B Loss Functions in EmbedTextNet

In Figure 3, we show the loss graphs of EmbedTextNet in the sentence similarity task. All the losses are converged well even if weight value is applied on reconstruction loss and correlation loss is added in VAE model.

C Databases

In Table 12, we show the summary of each dataset with its downstream task.

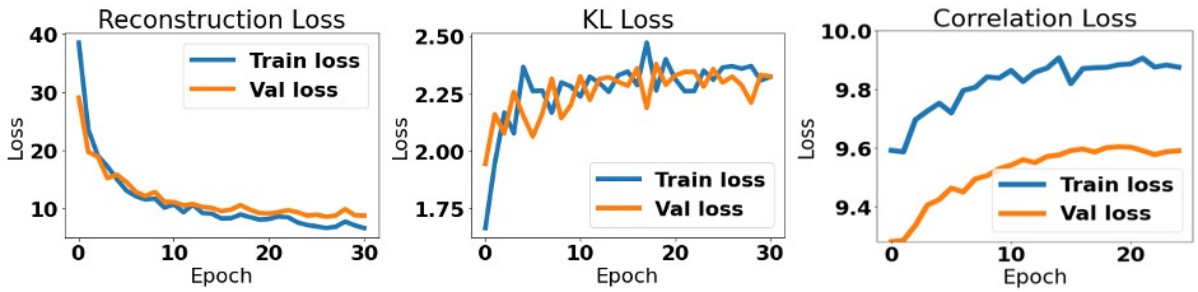

Figure 3: Loss functions in EmbedTextNet. The graphs are based on dimensionality reduction on sentence embedding (i.e. SBERT-1024D → 128D). Even if we increase the epochs, the losses are converged as this figure.

D Detailed Results about Ablation Study

Tables 13 - 18 showcase the detail experimental results of ablation study in each database reported in the paper.

E Additional Results for Text Similarity

Tables 19 and 20 cover the additional experimental results for text similarity tasks.

F Links for considered Datasets and Models

- Word Similarity Dataset: https://github.com/mfaruqui/eval-word-vectors. MIT License.
- Sentence Similarity Dataset: https://ixa2.si.ehu.eus/stswiki/index.php/Main_Page.
- M-Sentence Similarity Dataset: https://alt.qcri.org/semeval2017/task1/, https://www.sbert.net/examples/training/multilingual/README.html.
- Language Modelling Dataset: https://blog.salesforceairesearch.com/the-wikitext-long-term-dependency-language-modeling-dataset/. Creative Commons Attribution-ShareAlike License.
- Text Retrieval Dataset: https://archive.ics.uci.edu/ml/datasets/bag+of+words, https://github.com/erikbern/ann-benchmarks/. MIT License.
- PCA: https://scikit-learn.org/stable/modules/generated/sklearn.decomposition.PCA.html. BSD-3-Clause License.
- Algo: https://github.com/vyraun/Half-Size.
- UMAP: https://umap-learn.readthedocs.io/en/latest/index.html. BSD-3-Clause License.
- GloVe: https://nlp.stanford.edu/projects/glove/. Apache-2.0 License.
- SBERT, mSBERT: https://www.sbert.net. Apache-2.0 License.
- LSTM for Language Modelling: https://github.com/pytorch/examples/tree/main/word_language_model. BSD-3-Clause License.
- Faiss: https://github.com/facebookresearch/faiss. MIT License.

Table 13: SRC (ρ x 100) according to the different γ in reconstruction loss when GloVe-50D $\rightarrow$ 10D.

γ	YP	VERB	SimLex	353-SIM	TR-3K	MTurK-771	353-ALL	MTurk-287	RW	RG-65	MC-30	353-REL	Average
0.3	**16.47**	26.72	**20.5**	**48.2**	**51.94**	**38.09**	**38.88**	54.13	24.52	**49.7**	**44**	**33.01**	**37.18**
0.5	12.53	**30.88**	20.06	45.86	51.02	37.06	38.25	**57.74**	24.19	45.42	36.65	30.51	35.85
1	9.79	24.56	18.5	46.15	48.61	36.52	36.85	51.07	**24.67**	45.57	43.73	31.17	34.77

Table 14: SRC (ρ x 100) according to the different γ in reconstruction loss when SBERT-1024D $\rightarrow$ 16D.

γ	STS12	STS13	STS14	STS15	STS16	STS-B	SICK-R	Average
0.3	**59.03**	**66.58**	67.35	71.16	**68.70**	73.80	**72.55**	**68.45**
0.5	58.8	65.5	67.2	**71.2**	67.9	**74.22**	72.3	68.16
1	59.01	66.47	**67.45**	70.99	68.27	72.97	72.03	68.17

Table 15: SRC (ρ x 100) according to the different γ in reconstruction loss when mSBERT-768D $\rightarrow$ 16D.

γ	AR-EN	ES-EN	EN-DE	EN-TR	FR-EN	IT-EN	NL-EN	Average
3	60.42	65.09	67.24	53.19	62.43	64.31	66.15	62.69
5	70.66	71.64	68.43	65.52	66.23	69.89	70.29	68.95
10	**73.55**	**79.56**	**75.67**	**70.71**	**75.82**	**77.42**	**75.36**	**75.44**

Table 16: SRC (ρ x 100) in word similarity according to the different weighted reconstruction loss. Word embedding is decreased from GloVe-300D $\rightarrow$ 150D. S. Hinge and MAE mean squared hinge and mean absolute error.

Loss	YP	VERB	SimLex	353-SIM	TR-3K	MTurK-771	353-ALL	MTurk-287	RW	RG-65	MC-30	353-REL	Average
MSE	52.82	34.05	**38.59**	**73.26**	**75.47**	**65.56**	**65.52**	**65.51**	**41.2**	79.49	**78.91**	**59.98**	**60.86**
Huber	48.38	30.33	35.47	67.93	73.4	63.29	59.77	64.27	37.29	**80.06**	78.18	52.92	57.61
S.Hinge	55.76	33.16	37.12	65.99	72.38	61.09	60.04	60.67	37.73	78.82	78.89	55.35	58.08
MAE	**56.73**	**35.08**	37.49	63.67	72.89	64.16	58.93	63.09	39.42	76.82	70.37	57.36	58.00

Table 17: Detailed results (ρ x 100) about the effect of KL divergence (i.e. VAE) and modified reconstruction loss when GloVe-300D $\rightarrow$ 150D. Here, modified reconstruction loss means the including weighting value, W, introduced in Algorithm 1.

Type	YP	VERB	SimLex	353-SIM	TR-3K	MTurK-771	353-ALL	MTurk-287	RW	RG-65	MC-30	353-REL	Average
AE	51.29	29.01	35.28	64.22	72.03	61.04	56.26	60.1	39.1	**79.7**	**74.55**	50	56.05
AE with modified reconstruction loss	**51.53**	32.17	34.83	64.76	72.36	61.12	57.71	60.47	**40.16**	79.48	74.17	53.56	56.86
VAE with modified reconstruction loss	48.79	**34.93**	**35.66**	**68.79**	**74.47**	**62.92**	**60.68**	**63.78**	38.57	77.4	73.53	**55.89**	**57.95**

Table 18: Detailed results (ρ x 100) about other approaches and selected EmbedTextNet when mSBERT-768D $\rightarrow$ 16D. Correlation loss threshold N is explained in Algorithm 1.

Type	AR-EN	ES-EN	EN-DE	EN-TR	FR-EN	IT-EN	NL-EN	Average
β 5-VAE	-0.58	3.48	-2.17	1.38	-4.92	2.58	5.17	0.71
β 20-VAE	6.30	0.60	4.09	6.93	3.08	2.85	1.84	3.67
β 100-VAE	1.80	-0.91	3.36	-2.92	-0.05	6.37	2.07	1.39
Weighting manually more (x5) on reconstruction loss	70.93	76.35	75.06	69.98	75.08	77.02	**76.10**	74.36
Weighting manually more (x10) on reconstruction loss	72.51	77.10	75.00	70.62	73.71	75.40	75.80	74.30
VAE with cosine-similarity	71.36	77.09	74.60	69.82	74.11	76.05	75.43	74.06
EmbedTextNet, $N = 1$	71.7	77.71	**76.5**	69.54	**76.01**	**77.48**	76.08	75.00
EmbedTextNet, $N = 5$	**73.55**	**79.56**	75.67	**70.71**	75.82	77.42	75.36	**75.44**
EmbedTextNet, $N = 15$	72.29	77.33	74.49	70.24	75.15	77.45	75.28	74.60
EmbedTextNet, $N = 25$	72.00	77.24	74.14	70.05	74.85	77.25	75.63	74.45

Table 19: Additional results for word similarity (ρ x 100 (STD)) where embedding is decreased from GloVe-300D $\rightarrow$ 100D. Bold means the best result among dimensionality reductions.

Embeddings	YP	VERB	SimLex	353-SIM	TR-3K	MTurK-771	353-ALL	MTurk-287	RW	RG-65	MC-30	353-REL	Average
GloVe-300D	56.13	30.51	37.05	66.38	73.75	65.01	60.54	63.32	41.18	76.62	70.26	57.26	58.17 (0)
GloVe-200D	52.21	28.45	34.03	62.91	71.01	62.12	57.42	61.99	38.95	71.26	66.56	54.48	55.12 (0)
PCA-100D	36.40	29.28	25.15	48.92	60.00	48.35	42.08	53.48	25.28	62.48	59.96	36.35	43.98 (1.16)
Algo-100D	46.97	31.94	33.65	**68.99**	70.8	59.4	**63.36**	56.49	**38.93**	69.82	65.6	56.6	55.21 (0.54)
UMAP-100D	17.17	23.06	24.69	36.66	35.26	29.52	30.24	33.94	9.61	40.51	27.93	23.36	27.66 (2)
EmbedTextNet -100D	**47.1**	**32.88**	**34.87**	66.7	**71.15**	**60.36**	60.78	**62.45**	37.13	**76.06**	**70.56**	**56.9**	**56.41 (0.42)**

Table 20: Detailed performances for sentence similarity (ρ x 100 (STD)) where embedding is decreased from SBERT-1024D $\rightarrow$ 128D and SBERT-1024D $\rightarrow$ 16D. Bold describes the best result among dimensionality reductions. The measurements of Avg. GloVe and BERT are done using SentEval.

Embeddings	**STS12**	**STS13**	**STS14**	**STS15**	**STS16**	**Average**
SBERT-1024D	66.83	71.46	74.31	78.26	75.72	73.32 (0)
SBERT-768D	63.79	69.34	72.94	75.16	73.27	70.9 (0)
PCA-128D	64.76	71.32	73.56	78.15	74.87	72.53 (0.05)
Algo-128D	67.45	71.74	75.1	78.22	76.17	73.74 (0.06)
UMAP-128D	36.02	46.89	39.83	41.16	37.74	40.33 (0.12)
EmbedTextNet-128D	**67.75**	**75.12**	**76.11**	**81.15**	**77.15**	**75.46 (0.05)**
PCA-16D	57.64	60.33	64.75	68.47	65.95	63.43 (0.09)
Algo-16D	57.73	54.71	64.15	61.99	64.32	60.58 (0.1)
UMAP-16D	36	45.74	39.34	42.08	40.18	40.67 (0.88)
EmbedTextNet-16D	**59.03**	**66.58**	**67.35**	**71.16**	**68.70**	**66.56 (0.07)**
Avg. GloVe-300D	53.28	50.76	55.63	59.22	57.88	55.35 (0)
Avg. BERT-768D	50.05	52.91	54.91	63.37	64.94	57.24 (0)

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 6

☑ A2. Did you discuss any potential risks of your work?
Section 6

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 1

☒ A4. Have you used AI writing assistants when working on this paper?
We do not use the AI writing assistants.

B ☑ Did you use or create scientific artifacts?

Section 3, 4 and Appendix A, C, F

☑ B1. Did you cite the creators of artifacts you used?
Section 3, 4 and Appendix F

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Appendix F

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Appendix A, F

☐ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
Not applicable. Left blank.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 4 and Appendix C

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 4 and Appendix C

C ☑ Did you run computational experiments?

Section 4 and Appendix D, E

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Appendix A

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 4 and Appendix A

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 4 and Appendix D, E

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 4 and Appendix A

D ☒ Did you use human annotators (e.g., crowdworkers) or research with human participants?

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.