Exploiting Invertible Decoders for Unsupervised Sentence Representation Learning

Shuai Tang **Virginia R. de Sa**
Department of Cognitive Science, Halıcıoğlu Data Science Institute, UC San Diego
{shuaitang93,desa}@ucsd.edu

Abstract

Encoder-decoder models for unsupervised sentence representation learning using the distributional hypothesis effectively constrain the learnt representation of a sentence to only that needed to reproduce the next sentence. While the decoder is important to constrain the representation, these models tend to discard the decoder after training since only the encoder is needed to map the input sentence into a vector representation. However, parameters learnt in the decoder also contain useful information about the language. In order to utilise the decoder after learning, we present two types of decoding functions whose inverse can be easily derived without expensive inverse calculation. Therefore, the inverse of the decoding function can serve as another encoder that produces sentence representations. We show that, with careful design of the decoding functions, the model learns good sentence representations, and the ensemble of the representations produced from the encoder and the inverse of the decoder demonstrate even better generalisation ability and solid transferability.

1 Introduction

Learning sentence representations from unlabelled data is becoming increasingly prevalent in both the machine learning and natural language processing research communities, as it efficiently and cheaply allows knowledge extraction that can successfully transfer to downstream tasks. Methods built upon the distributional hypothesis (Harris, 1954) and distributional similarity (Firth, 1957) can be roughly categorised into two types:

Word-prediction Objective: This objective pushes the system to make better predictions of words in a given sentence. As the nature of the objective is to predict words, these are also called generative models. In one of the two classes of models of this type, an encoder-decoder model is learnt using a corpus of contiguous sentences (Kiros et al., 2015; Gan et al., 2017; Tang et al., 2018) to make predictions of the words in the next sentence given the words in the current one. After training, the decoder is usually discarded as it is only needed during training and is not designed to produce sentence representations. In the other class of models of this type, a large language model is learnt (Peters et al., 2018; Radford et al., 2018; Devlin et al., 2018) on unlabelled corpora, which could be an autoregressive model or a masked language model, which gives extremely powerful language encoders but requires massive computing resources and training time.

Similarity-based Objective: The objective here relies on a predefined similarity function to force the model to produce more similar representations for adjacent sentences than those that are not (Li and Hovy, 2014; Jernite et al., 2017; Nie et al., 2017; Logeswaran and Lee, 2018). Therefore, the inductive biases introduced by the two key components, the differential similarity function and the context window, in the objective crucially determine the quality of learnt representations and what sentence information can be encoded in them.

To avoid tuning the inductive biases in the similarity-based objective, we follow the word-prediction objective with an encoder and a decoder, and we are particularly interested in exploiting invertible decoding functions, which can then be used as additional encoders during testing. The contribution of our work is summarised as follows:

1. The decoder is used in testing to produce sentence representations. With careful design, the inverse function of the decoder is easy to derive with no expensive inverse calculation.

2. The inverse of the decoder provides high-quality sentence representations as well as

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 4050–4060
Florence, Italy, July 28 - August 2, 2019.

the encoder and, since the inverse function of the decoder naturally behaves differently from the encoder, the representations from both functions complement each other and an ensemble of both provides good results on downstream tasks.

3. The analyses show that the effectiveness of the invertible constraint enforced on the decoder side and learning from unlabelled corpora help the produced representations to better capture the meaning of sentences.

2 Related Work

Learning vector representations for words with a word embedding matrix as the encoder and a context word embedding matrix as the decoder (Mikolov et al., 2013a; Lebret and Collobert, 2014; Pennington et al., 2014; Bojanowski et al., 2017) can be considered as a word-level example of our approach, as the models learn to predict the surrounding words in the context given the current word, and the context word embeddings can also be utilised to augment the word embeddings (Pennington et al., 2014; Levy et al., 2015). We are thus motivated to explore the use of sentence decoders after learning instead of ignoring them as most sentence encoder-decoder models do.

Our approach is to invert the decoding function in order to use it as another encoder to assist the original encoder. In order to make computation of the inverse function well-posed and tractable, careful design of the decoder is needed. A simple instance of an invertible decoder is a linear projection with an orthonormal square matrix, whose transpose is its inverse. A family of bijective transformations with non-linear functions (Dinh et al., 2014; Rezende and Mohamed, 2015; Kingma et al., 2016) can also be considered as it empowers the decoder to learn a complex data distribution.

In our paper, we exploit two types of plausible decoding functions, including linear projection and bijective functions with neural networks (Dinh et al., 2014), and with proper design, the inverse of each of the decoding functions can be derived without expensive inverse calculation after learning. Thus, the decoder function can be utilised along with the encoder for building sentence representations. We show that the ensemble of the encoder and the inverse of the decoder outperforms each of them.

3 Model Design

Our model has similar structure to that of skip-thought (Kiros et al., 2015) and, given the neighbourhood hypothesis (Tang et al., 2017), learns to decode the next sentence given the current one instead of predicting both the previous sentence and the next one at the same time.

3.1 Training Objective

We have previously shown (Tang et al., 2018) that neither an autoregressive nor an RNN decoder is necessary for learning sentence representations that excel on downstream tasks. As the autoregressive decoders are slow to train and the quality of the generated sequences is not highly correlated with that of the representations of the sentences, our model only learns to predict words in the next sentence in a non-autoregressive fashion.

Suppose that the i-th sentence $S_i = \{w_1, w_2, ..., w_{N_i}\}$ has N_i words, and S_{i+1} has N_{i+1} words. The learning objective is to maximise the averaged log-likelihood for all sentence pairs:

$$\ell_{S_{i+i}|S_i}(\boldsymbol{\phi}, \boldsymbol{\theta}) = \frac{1}{N_{i+1}} \sum_{w_j \in S_{i+1}} \log P(w_j|S_i)$$

where $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ contain the parameters in the encoder $f_{\text{en}}(S_i; \boldsymbol{\theta})$ and the decoder $f_{\text{de}}(\boldsymbol{z}_i; \boldsymbol{\phi})$ respectively. The forward computation of our model for a given sentence pair $\{S_i, S_{i+1}\}$, in which the words in S_i are the input to the learning system and the words in S_{i+1} are targets is defined as:

$$\boldsymbol{z}_i = f_{\text{en}}(S_i; \boldsymbol{\theta})$$
$$\boldsymbol{x}_i = f_{\text{de}}(\boldsymbol{z}_i; \boldsymbol{\phi})$$

where $\boldsymbol{z}_i$ is the vector representation of S_i, and $\boldsymbol{x}_i$ is the vector output of the decoder which will be compared with the vector representations of words in the next sentence S_{i+1}. Since calculating the likelihood of generating each word involves a computationally demanding softmax function, the negative sampling method (Mikolov et al., 2013a) is applied to replace the softmax, and $\log P(w_j|s_i)$ is calculated as:

$$\log \sigma(\boldsymbol{x}_i^\top \boldsymbol{v}_{w_j}) + \sum_{k=1}^{K} \mathbb{E}_{w_k \sim P_e(w)} \log \sigma(-\boldsymbol{x}_i^\top \boldsymbol{v}_{w_k})$$

where $\boldsymbol{v}_{w_k} \in \mathbb{R}^{d_{\boldsymbol{v}}}$ is the pretrained vector representation for w_k, the empirical distribution $P_e(w)$

is the unigram distribution of words in the training corpus raised to power 0.75 as suggested in prior work (Mikolov et al., 2013b), and K is the number of negative samples. In this case, we enforce the output of the decoder $\boldsymbol{x}_i$ to have the same dimensionality as the pretrained word vectors $\boldsymbol{v}_{w_j}$. The loss function is summed over all contiguous sentence pairs in the training corpus. For simplicity, we omit the subscription for indexing the sentences in the following sections.

3.2 Encoder

The encoder $f_{\text{en}}(S;\boldsymbol{\theta})$ is a bi-directional Gated Recurrent Unit (Chung et al., 2014) with d dimensions in each direction. It processes word vectors in an input sentence $\{\boldsymbol{v}_{w_1}, \boldsymbol{v}_{w_2}, ..., \boldsymbol{v}_{w_N}\}$ sequentially according to the temporal order of the words, and generates a sequence of hidden states. During learning, in order to reduce the computation load, only the last hidden state serves as the sentence representation $\boldsymbol{z} \in \mathbb{R}^{d_{\boldsymbol{z}}}$, where $d_{\boldsymbol{z}} = 2d$.

3.3 Decoder

As the goal is to reuse the decoding function $f_{\text{de}}(\boldsymbol{z})$ as another plausible encoder for building sentence representations after learning rather than ignoring it, one possible solution is to find the inverse function of the decoder function during testing, which is noted as $f_{\text{de}}^{-1}(\boldsymbol{x})$. In order to reduce the complexity and the running time during both training and testing, the decoding function $f_{\text{de}}(\boldsymbol{z})$ needs to be easily invertible. Here, two types of decoding functions are considered and explored.

3.3.1 Linear Projection

In this case, the decoding function is a linear projection, which is $\boldsymbol{x} = f_{\text{de}}(\boldsymbol{z}) = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}$, where $\boldsymbol{W} \in \mathbb{R}^{d_{\boldsymbol{v}} \times d_{\boldsymbol{z}}}$ is a trainable weight matrix and $\boldsymbol{b} \in \mathbb{R}^{d_{\boldsymbol{v}} \times 1}$ is the bias term.

As f_{de} is a linear projection, the simplest situation is when $\boldsymbol{W}$ is an orthogonal matrix and its inverse is equal to its transpose. Often, as the dimensionality of vector $\boldsymbol{z}$ doesn't necessarily need to match that of the word vectors $\boldsymbol{v}$, $\boldsymbol{U}$ is not a square matrix . To enforce invertibility on $\boldsymbol{W}$, a row-wise orthonormal regularisation on $\boldsymbol{W}$ is applied during learning, which leads to $\boldsymbol{W}\boldsymbol{W}^\top = \boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. Thus the inverse function is simply $\boldsymbol{z} = f_{\text{de}}^{-1}(\boldsymbol{x}) = \boldsymbol{W}^\top(\boldsymbol{x} - \boldsymbol{b})$, which is easily computed. The regularisation formula is $||\boldsymbol{W}\boldsymbol{W}^\top - \boldsymbol{I}||_F$, where $||\cdot||_F$ is the Frobenius norm.[1] Specifically, the update rule (Cissé et al., 2017) for the regularisation is:

$$\boldsymbol{W} := (1+\beta)\boldsymbol{W} - \beta(\boldsymbol{W}\boldsymbol{W}^\top)\boldsymbol{W}$$

The usage of the decoder during training and testing is defined as follows:

$$\text{Training: } \boldsymbol{x} = f_{\text{de}}(\boldsymbol{z}) = \boldsymbol{W}\boldsymbol{z} + \boldsymbol{b}$$
$$\text{Testing: } \boldsymbol{z} = f_{\text{de}}^{-1}(\boldsymbol{x}) = \boldsymbol{W}^\top(\boldsymbol{x} - \boldsymbol{b})$$

Therefore, the decoder is also utilised after learning to serve as a linear encoder in addition to the RNN encoder.

3.3.2 Bijective Functions

A general case is to use a bijective function as the decoder, as bijective functions are naturally invertible. However, the inverse of a bijective function could be hard to find and its calculation could also be computationally expensive.

A family of bijective transformations was designed in NICE (Dinh et al., 2014), and the simplest continuous bijective function $f : \mathbb{R}^D \to \mathbb{R}^D$ and its inverse f^{-1} is defined as:

$$\begin{aligned} h: &\quad \boldsymbol{y}_1 = \boldsymbol{x}_1, &\quad \boldsymbol{y}_2 = \boldsymbol{x}_2 + m(\boldsymbol{x}_1) \\ h^{-1}: &\quad \boldsymbol{x}_1 = \boldsymbol{y}_1, &\quad \boldsymbol{x}_2 = \boldsymbol{y}_2 - m(\boldsymbol{y}_1) \end{aligned}$$

where $\boldsymbol{x}_1$ is a d-dimensional partition of the input $\boldsymbol{x} \in \mathbb{R}^D$, and $m : \mathbb{R}^d \to \mathbb{R}^{D-d}$ is an arbitrary continuous function, which could be a trainable multi-layer feedforward neural network with non-linear activation functions. The layer h is named as an 'additive coupling layer' (Dinh et al., 2014), which has unit Jacobian determinant. To allow the learning system to explore more powerful transformation, we follow the design of the '**affine** coupling layer' (Dinh et al., 2016):

$$\begin{aligned} h: &\quad \boldsymbol{y}_1 = \boldsymbol{x}_1, &\quad \boldsymbol{y}_2 = \boldsymbol{x}_2 \circ \exp(s(\boldsymbol{x}_1)) + t(\boldsymbol{x}_1) \\ h^{-1}: &\quad \boldsymbol{x}_1 = \boldsymbol{y}_1, &\quad \boldsymbol{x}_2 = (\boldsymbol{y}_2 - t(\boldsymbol{y}_1)) \circ \exp(-s(\boldsymbol{x}_1)) \end{aligned}$$

where $s : \mathbb{R}^d \to \mathbb{R}^{D-d}$ and $t : \mathbb{R}^d \to \mathbb{R}^{D-d}$ are both neural networks with linear output units, and $\circ$ is the Hadamard product.

The requirement of the continuous bijective transformation is that, the dimensionality of the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ need to match exactly.

[1] As often the dimension of sentence vectors are equal to or larger than that of word vectors, $\boldsymbol{W}$ has more columns than rows. If this is not the case, then the regulariser becomes $||\boldsymbol{W}^\top\boldsymbol{W} - \boldsymbol{I}||_F$.

In our case, the output $\boldsymbol{x} \in \mathbb{R}^{d_v}$ of the decoding function f_{de} has lower dimensionality than the input $\boldsymbol{z} \in \mathbb{R}^{d_z}$ does. Our solution is to add an orthonormal regularised linear projection before the bijective function to transform the vector representation of a sentence to the desired dimension.

The usage of the decoder that is composed of a bijective function and a regularised linear projection during training and testing is defined as:

$$\text{Training: } \boldsymbol{x} = f_{\text{de}}(\boldsymbol{z}) = h(\boldsymbol{W}\boldsymbol{z} + \boldsymbol{b})$$
$$\text{Testing: } \boldsymbol{z} = f_{\text{de}}^{-1}(\boldsymbol{x}) = \boldsymbol{W}^{\top}(h^{-1}(\boldsymbol{x}) - \boldsymbol{b})$$

3.4 Using Decoder in the Test Phase

As the decoder is easily invertible, it is also used to produce vector representations. The post-processing step (Arora et al., 2017) that removes the top principal component is applied on the representations from f_{en} and f_{de}^{-1} individually. In the following sections, $\boldsymbol{z}_{\text{en}}$ denotes the post-processed representation from f_{en}, and $\boldsymbol{z}_{\text{de}}$ from f_{de}^{-1}. Since f_{en} and f_{de}^{-1} naturally process sentences in distinctive ways, it is reasonable to expect that the ensemble of $\boldsymbol{z}_{\text{en}}$ and $\boldsymbol{z}_{\text{de}}$ will outperform each of them.

4 Experimental Design

Experiments are conducted in PyTorch (Paszke et al., 2017), with evaluation using the SentEval package (Conneau et al., 2017) with modifications to include the post-processing step. Word vectors $\boldsymbol{v}_{w_j}$ are initialised with FastText (Bojanowski et al., 2017), and fixed during learning.

4.1 Unlabelled Corpora

Two unlabelled corpora, including **B**ookCorpus (Zhu et al., 2015) and **U**MBC News Corpus (Han et al., 2013), are used to train models with invertible decoders. These corpora are referred to as **B**, and **U** in Tables 3 and 5. The UMBC News Corpus is roughly twice as large as the BookCorpus, and the details are shown in Table 1.

Name	# of sentences
BookCorpus (**B**)	74 million
UMBC News (**U**)	134.5 million

Table 1: **Summary statistics** of the two corpora used. For simplicity, the two corpora are referred to as **B** and **U** in the following tables respectively.

4.2 Evaluation Tasks

4.3 Unsupervised Evaluation

The unsupervised tasks include five tasks from SemEval Semantic Textual Similarity (STS) in 2012-2016 (Agirre et al., 2015, 2014, 2016, 2012, 2013) and the SemEval2014 Semantic Relatedness task (SICK-R) (Marelli et al., 2014).

The cosine similarity between vector representations of two sentences determines the textual similarity of two sentences, and the performance is reported in Pearson's correlation score between human-annotated labels and the model predictions on each dataset.

4.4 Supervised Evaluation

Supervised evaluation tasks include Semantic relatedness (SICK) (Marelli et al., 2014), SemEval (STS-B) (Cer et al., 2017), paraphrase detection (MRPC) (Dolan et al., 2004), question-type classification (TREC) (Li and Roth, 2002), movie review sentiment (MR) (Pang and Lee, 2005), Stanford Sentiment Treebank (SST) (Socher et al., 2013), customer product reviews (CR) (Hu and Liu, 2004), subjectivity/objectivity classification (SUBJ) (Pang and Lee, 2004), opinion polarity (MPQA) (Wiebe et al., 2005).

In these tasks, MR, CR, SST, SUBJ, MPQA and MRPC are binary classification tasks; TREC is a multi-class classification task. SICK and MRPC require the same feature engineering method (Tai et al., 2015) in order to compose a vector from vector representations of two sentences to indicate the difference between them.

4.5 Hyperparameter Tuning

The hyperparameters are tuned on the averaged scores on STS14 of the model trained on **B**ookCorpus, thus these results are marked with a * in Table 3 to indicate potential overfitting. These parameters are then used for all tasks.

The hyperparameter settings for our model are summarised as follows: the batch size $N = 512$, the dimension of sentence vectors $d_{\boldsymbol{z}} = 2048$, the dimension of word vectors $d_{\boldsymbol{v}_{w_j}} = 300$, the number of negative samples $K = 5$, and the initial learning rate is 5×10^{-4} which is kept fixed during learning. The Adam optimiser (Kingma and Ba, 2014) with gradient clipping (Pascanu et al., 2013) is applied for stable learning. Each model in our experiment is only trained for one epoch on the given training corpus.

Toronto BookCorpus	Hrs	Unsupervised tasks: Avg of STS tasks (STS12-16, SICK14)	Supervised tasks: Avg of SICK-R, STS-B	Supervised tasks: Avg of Binary-CLS tasks (MR, CR, SUBJ, MPQA, SST)	Supervised tasks: MRPC
Generative Objective with **Invertible** Linear Projection					
z_{en}	3	64.9	82.2	86.2	75.1/83.4
z_{de}		67.6	82.2	85.2	73.7/82.5
ensemble(z_{en}, z_{de})		70.2	83.1	87.0	76.5/83.7
Generative Objective with Linear Projection					
z_{en}	3	54.6 (↓10.3)	79.5 (↓2.7)	85.6 (↓0.6)	75.1/82.5
z_{de}		69.5 (↑1.9)	82.4 (↑0.2)	84.7 (↓0.5)	74.3/82.0
ensemble(z_{en}, z_{de})		66.9 (↓3.3)	82.8 (↓0.3)	86.3 (↓0.7)	76.2/83.5
Generative Objective with Bijective Transformation + **Invertible** Linear Projection					
z_{en}	3.3	67.1	82.1	85.4	74.3/82.2
z_{de}		67.6	82.1	85.0	74.6/82.4
ensemble(z_{en}, z_{de})		70.0	82.9	86.5	76.2/83.0
Generative Objective with Bijective Transformation + Linear Projection					
z_{en}	3.3	63.4 (↓3.7)	81.7 (↓0.4)	85.2 (↓0.2)	76.9/84.3
z_{de}		67.8 (↑0.2)	82.2 (↑0.1)	84.1 (↓0.9)	74.7/82.0
ensemble(z_{en}, z_{de})		69.4 (↓0.6)	82.5 (↓0.4)	86.1 (↓0.4)	76.6/83.4

Table 2: **The effect of the invertible constraint on linear projection**. The arrow and its associated value of a representation is the relative performance gain or loss compared to its comparison partner with the invertible constraint. As shown, the invertible constraint does help improve each representation, and ensures the ensemble of two encoding functions gives better performance. Better view in colour.

For the linear projection, β in the invertible constraint is set to be 0.01, and after learning, all 300 eigenvalues are close to 1. For the bijective transformation, in order to make sure that each output unit is influenced by all input units, we stack four affine coupling layers in the bijective transformation (Dinh et al., 2014). The non-linear mappings s and t are both neural networks with one hidden layer with rectified linear activation function.

4.6 Representation Pooling

Various pooling functions are applied to produce vector representations for input sentences.

For unsupervised evaluation tasks, as recommended in previous studies (Pennington et al., 2014; Kenter et al., 2016; Wieting and Gimpel, 2017), a global mean-pooling function is applied on both the output of the RNN encoder f_{en} to produce a vector representation z_{en} and the inverse of the decoder f_{de}^{-1} to produce z_{de}.

For supervised evaluation tasks, three pooling functions, including global max-, min-, and mean-pooling, are applied on top of the encoder and the outputs from three pooling functions are concatenated to serve as a vector representation for a given sentence. The same representation pooling strategy is applied on the inverse of the decoder.

The reasons for applying different representation pooling strategies for the two categories of tasks include:

(1) cosine similarity of two vector representations is directly calculated in unsupervised evaluation tasks to determine the textual similarity of two sentences; it suffers from the curse-of-dimensionality (Donoho, 2000), which leads to more equidistantly distributed representations for higher dimensional vector representations decreasing the difference among similarity scores.

(2) given Cover's theorem (Cover, 1965) and the blessings-of-dimensionality property, it is more likely for the data points to be linearly separable when they are presented in high dimensional space, and in the supervised evaluation tasks, high dimensional vector representations are preferred as a linear classifier will be learnt to evaluate how well the produced sentence representations are linearly separable;

(3) in our case, both the encoder and the inverse of the decoder are capable of producing a vector representation per time step in a given sentence;

[1]Arora et al. (2017);[2]Wieting et al. (2015);[3]Wieting and Gimpel (2018);[4]Conneau et al. (2017);
[5]Wieting and Gimpel (2018);[6−10]Agirre et al. (2012, 2013, 2014, 2015, 2016);
[11]Marelli et al. (2014);[12]Mikolov et al. (2017)

Task	Un. Training						Semi.		Su.	
	Linear		**Bijective**		fastText		[2]PSL		[4]Infer	[3]ParaNMT
	B	**U**	**B**	**U**	[12]avg	[1]WR	[1]avg	[1]WR	Sent	(concat.)
[6]STS12	61.5	61.3	60.8	**62.7**	58.3	58.8	52.8	59.5	58.2	67.7
[7]STS13	61.3	61.8	60.7	**62.2**	51.0	59.9	46.4	61.8	48.5	62.8
[8]STS14	⋆71.6	72.1	72.1	**73.2**	65.2	69.4	59.5	73.5	67.1	76.9
[9]STS15	76.1	76.9	76.6	**77.6**	67.7	74.2	60.0	76.3	71.1	79.8
[10]STS16	74.8	76.1	75.8	**76.9**	64.3	72.4	-	-	71.2	76.8
[11]SICK14	**76.1**	73.6	74.2	73.9	69.8	72.3	66.4	72.9	73.4	-
Average	70.2	70.3	70.0	**71.1**	62.7	67.8	-	-	64.9	-

Table 3: **Results on unsupervised evaluation tasks** (Pearson's $r \times 100$) . **Bold** numbers are the best results among unsupervised transfer models, and underlined numbers are the best ones among all models. 'WR' refers to the post-processing step that removes the top principal component.

although during training only the last one is regarded as the sentence representation for fast training speed, it is more reasonable to make use of all representations at all time steps with various pooling functions to compute a vector representation to produce high-quality sentence representations that excel on the downstream tasks.

5 Discussion

It is worth discussing the motivation of the model design and the observations in our experiments. As mentioned as one of the take-away messages in previous work (Wieting and Kiela, 2019), to demonstrate the effectiveness of the invertible constraint, the comparison of our model with the constraint and its own variants use the same word embeddings from FastText (Bojanowski et al., 2017) and have the same dimensionaility of sentence representations during learning, and use the same classifier on top of the produced representations with the same hyperparameter settings.

Overall, given the performance of the inverse of each decoder (linear and bijective function) presented in Tables 3 and 5, it is reasonable to state that the inverse of each decoder provides high-quality sentence representations as well as the encoder. We found no significant difference between the two decoders in terms of the performance on the downstream tasks. In this section, observations and thoughts are presented based on the analyses of our models with the linear invertible constraint.

5.1 Effect of Invertible Constraint

The motivation of enforcing the invertible constraint on the decoder during learning is to make it usable and potentially helpful during testing in terms of boosting the performance of the lone RNN encoder in the encoder-decoder models (instead of ignoring the decoder part after learning). Therefore, it is important to check the necessity of the invertible constraint on the decoders.

A model with the same hyperparameter settings but without the invertible constraint is trained as the baseline model, and macro-averaged results that summarise the same type of tasks are presented in Table 2.

As noted in the prior work (Hill et al., 2016), there exists significant inconsistency between the group of unsupervised tasks and the group of supervised ones, it is possible for a model to excel on one group of tasks but fail on the other one. As presented in our table, the inverse of the decoder tends to perform better than the encoder on unsupervised tasks, and the situation reverses when it comes to the supervised ones.

In our model, the invertible constraint **helps the RNN encoder** f_{en} to perform better on the unsupervised evaluation tasks, and **helps the inverse of the decoder** f_{de}^{-1} to provide better results on single sentence classification tasks. An interesting observation is that, by enforcing the invertible constraint, the model learns to sacrifice the performance of f_{de}^{-1} and improve the performance of f_{en} on unsupervised tasks to mitigate the gap be-

tween the two encoding functions, which leads to more aligned vector representations between f_{en} and f_{de}^{-1}.

5.2 Effect of Ensemble

Although encouraging the invertible constraint leads to slightly poorer performance of f_{de}^{-1} on unsupervised tasks, it generally leads to better sentence representations when the ensemble of the encoder f_{en} and the inverse of the decoder f_{de}^{-1} is considered. Specifically, for unsupervised tasks, the ensemble is an average of the two vector representations produced from the two encoding functions during the testing time, and for supervised tasks, the concatenation of the two representations is regarded as the representation of a given sentence. The ensemble method is recommended in prior work (Pennington et al., 2014; Levy et al., 2015; Wieting and Gimpel, 2017; McCann et al., 2017; Tang et al., 2018; Wieting and Kiela, 2019).

As presented in Table 2, on unsupervised evaluation tasks (STS12-16 and SICK14), the ensemble of two encoding functions is averaging, which benefits from aligning representations from f_{en} and f_{de}^{-1} by enforcing the invertible constraint. While in the learning system without the invertible constraint, the ensemble of two encoding functions provides worse performance than f_{de}^{-1}.

On supervised evaluation tasks, as the ensemble method is concatenation and a linear model is applied on top of the concatenated representations, as long as the two encoding functions process sentences distinctively, the linear classifier is capable of picking relevant feature dimensions from both encoding functions to make good predictions, thus there is no significant difference between our model with and without invertible constraint.

5.3 Effect of Learning

Recent research (Wieting and Kiela, 2019) showed that the improvement on supervised evaluation tasks obtained by learning from labelled or unlabelled corpora is rather insignificant compared to random initialised projections on top of pretrained word vectors. Another interesting direction of research that utilises probabilistic random walk models on the unit sphere (Arora et al., 2016, 2017; Ethayarajh, 2018) derived several simple yet effective post-processing methods that operate on pretrained word vectors and are able to boost the performance of the averaged word vectors as the sentence representation on unsupervised tasks. While these papers reveal interesting aspects of the downstream tasks and question the need for optimising a learning objective, our results show that learning on unlabelled corpora helps.

On **unsupervised** evaluation tasks, in order to show that learning from an unlabelled corpus helps, the performance of our learnt representations should be directly compared with the pretrained word vectors, FastText in our system, at the same dimensionality with the same postprocessing (Arora et al., 2017). The word vectors are scattered in the 300-dimensional space, and our model has a decoder that is learnt to project a sentence representation $\boldsymbol{z} \in \mathbb{R}^{d_{\boldsymbol{z}}}$ to $\boldsymbol{x} = f_{\text{de}}(\boldsymbol{z}; \boldsymbol{\phi}) \in \mathbb{R}^{300}$. The comparison of our learnt representations with averaged word vectors with the same postprocessing are presented in Table 4.

As shown in the Table 4, the performance of our learnt system is better than FastText at the same dimensionality. It is worth mentioning that, in our system, the final representation is an average of postprocessed word vectors and the learnt representations $\boldsymbol{x}$, and the invertible constraint guarantees that the ensemble of both gives better performance. Otherwise, as discussed in the previous section, an ensemble of postprocessed word vectors and some random encoders won't necessarily lead to stronger results. Table 3 also provides evidence for the effectiveness of learning on the unsupervised evaluation tasks.

Task	Linear	Bijective	FastText+WR
STS12	60.7	**60.9**	58.8
STS13	**61.1**	60.0	59.9
STS14	**71.7**	**71.7**	69.4
STS15	**75.9**	75.4	74.2
STS16	**74.9**	73.5	72.4
SICK14	75.7	**75.8**	72.3
Average	**70.0**	69.6	67.8

Table 4: **Comparison** of the learnt representations in our system with the **same dimensionality** as the average of the same pretrained word vectors on unsupervised evaluation tasks. The encoding function that is learnt to compose a sentence representation from pretrained word vectors outperforms averaging the same word vectors, which supports our argument that learning helps to produce higher-quality sentence representations.

On **supervised** evaluation tasks, we agree that

[1]Conneau et al. (2017);[2]Hill et al. (2016); [3]Kiros et al. (2015);[4]Ba et al. (2016);[5]Gan et al. (2017);
[6]Jernite et al. (2017);[7]Nie et al. (2017);[8]Zhao et al. (2015);[9]Logeswaran and Lee (2018);[10]Marelli et al. (2014);
[11]Dolan et al. (2004);[12]Li and Roth (2002);[13]Pang and Lee (2005);[14]Hu and Liu (2004)
[15]Pang and Lee (2004);[16]Wiebe et al. (2005);[17]Socher et al. (2013);[18]Wieting and Kiela (2019)

Model	Hrs	[10]SICK-R	[10]SICK-E	[11]MRPC	[12]TREC	[13]MR	[14]CR	[15]SUBJ	[16]MPQA	[17]SST
Supervised task-dependent training - No transfer learning										
[8]AdaSent	-	-	-	-	92.4	83.1	86.3	95.5	93.3	-
[1]TF-KLD	-	-	-	80.4/85.9	-	-	-	-	-	-
Supervised training - Transfer learning										
[1]InferSent	<24	88.4	86.3	76.2/83.1	88.2	81.1	86.3	92.4	90.2	84.6
Unsupervised training with ordered sentences										
[2]FastSent+AE	2	-	-	71.2/79.1	80.4	71.8	76.5	88.8	81.5	-
[4]ST+LN	720	85.8	79.5	-	88.4	79.4	83.1	93.7	89.3	82.9
[5]CNN-LSTM	-	86.2	-	76.5/83.8	92.6	77.8	82.1	93.6	89.4	-
[6]DiscSent	8	-	-	75.0/ -	87.2	-	-	93.0	-	-
[7]DisSent	-	79.1	80.3	- / -	84.6	82.5	80.2	92.4	89.6	82.9
[9]MC-QT	11	86.8	-	76.9/**84.0**	**92.8**	80.4	**85.2**	93.9	89.4	-
B - Bijective z	3.3	87.9	84.5	76.2/83.0	89.6	80.3	82.6	94.6	89.3	85.6
B - Linear z	3	**88.1**	85.2	76.5/83.7	90.0	**81.3**	83.5	94.6	**89.5**	**85.9**
U - Bijective z	10	87.8	85.2	76.4/83.7	90.8	80.9	82.7	94.6	89.2	83.3
U - Linear z	8.8	87.8	**85.9**	**77.5**/83.8	92.2	**81.3**	83.4	**94.7**	**89.5**	**85.9**
No training - [18]Global max-pooling on top of random projection										
BOREP	0	85.9	84.3	73.7/ -	89.5	78.6	79.9	93.0	88.8	82.5
RandLSTM	0	86.6	83.0	74.7/ -	88.4	78.2	79.9	92.8	88.2	83.2
ESN	0	87.2	85.1	75.3/ -	92.2	79.1	80.2	93.4	88.9	84.6

Table 5: **Results on supervised evaluation tasks. Bold** numbers are the best results among unsupervised transfer models with ordered sentences, and underlined numbers are the best ones among all models.

higher dimensional vector representations give better results on the downstream tasks. Compared to random projections with 4096×6 output dimensions, learning from unlabelled corpora leverages the distributional similarity (Firth, 1957) at the sentence-level into the learnt representations and potentially helps capture the meaning of a sentence. In our system, the raw representations are in 2400-dimensional space, and the use of various pooling functions expands it to 2048×6 dimensions, which is half as large as the random projection dimension and still yields better performance. Both our models and random projections with no training are presented in Table 5.

The evidence from both sets of downstream tasks support our argument that learning from unlabelled corpora helps the representations capture meaning of sentences. However, current ways of incorporating the distributional hypothesis only utilise it as a weak and noisy supervision, which might limit the quality of the learnt sentence representations.

6 Conclusion

Two types of decoders, including an orthonormal regularised linear projection and a bijective transformation, whose inverses can be derived effortlessly, are presented in order to utilise the decoder as another encoder in the testing phase. The experiments and comparisons are conducted on two large unlabelled corpora, and the performance on the downstream tasks shows the high usability and generalisation ability of the decoders in testing.

Analyses show that the invertible constraint enforced on the decoder encourages the usual encoder and the invertible decoder to learn from the other one during learning, and provides improved encoding functions after learning. An ensemble of

the encoder and the inverse of the decoder gives even better performance when the invertible constraint is applied on the decoder side. Furthermore, by comparing with prior work, we argue that learning from unlabelled corpora indeed helps to improve the sentence representations, although the current way of utilising corpora might not be optimal.

We view our work as unifying the generative and discriminative objectives for unsupervised sentence representation learning, as the decoder is trained with a generative objective which when inverted can be seen as creating a discriminative target.

The proposed method in our implementation doesn't provide state-of-the-art performance on the downstream tasks, but we see our method as an opportunity to fuse all possible components in a model, even a usually discarded decoder, to produce sentence representations. Future work could potentially expand our work into an end-to-end invertible model that is able to produce high-quality representations by omnidirectional computations.

Acknowledgements

Many thanks to Andrew Ying for helpful clarifications on several concepts and also to Adobe gift funding and NSF IIS 1528214.

References

Eneko Agirre, Carmen Banea, Claire Cardie, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iñigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. Semeval-2015 task 2: Semantic textual similarity, english, spanish and pilot on interpretability. In *SemEval@NAACL-HLT*.

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*.

Eneko Agirre, Carmen Banea, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, Rada Mihalcea, German Rigau, and Janyce Wiebe. 2016. Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation. In *SemEval@NAACL-HLT*.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, and Aitor Gonzalez-Agirre. 2012. Semeval-2012 task 6: A pilot on semantic textual similarity. In *SemEval@NAACL-HLT*.

Eneko Agirre, Daniel M. Cer, Mona T. Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In **SEM@NAACL-HLT*.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2016. A latent variable model approach to pmi-based word embeddings. *TACL*, 4:385–399.

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2017. A simple but tough-to-beat baseline for sentence embeddings. In *International Conference on Learning Representations*.

Jimmy Ba, Ryan Kiros, and Geoffrey E. Hinton. 2016. Layer normalization. *CoRR*, abs/1607.06450.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *TACL*, 5:135–146.

Daniel M. Cer, Mona T. Diab, Eneko Agirre, Iñigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity multilingual and crosslingual focused evaluation. In *SemEval@ACL*.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*.

Moustapha Cissé, Piotr Bojanowski, Edouard Grave, Yann Dauphin, and Nicolas Usunier. 2017. Parseval networks: Improving robustness to adversarial examples. In *ICML*.

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

Thomas M Cover. 1965. Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. *IEEE transactions on electronic computers*, (3):326–334.

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

Laurent Dinh, David Krueger, and Yoshua Bengio. 2014. Nice: Non-linear independent components estimation. *CoRR*, abs/1410.8516.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. 2016. Density estimation using real nvp. *CoRR*, abs/1605.08803.

William B. Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*.

David L. Donoho. 2000. Aide-memoire . high-dimensional data analysis : The curses and blessings of dimensionality.

Kawin Ethayarajh. 2018. Unsupervised random walk sentence embeddings: A strong but simple baseline. In *Rep4NLP@ACL*.

J. R. Firth. 1957. A synopsis of linguistic theory.

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

Lushan Han, Abhay L Kashyap, Tim Finin, James Mayfield, and Jonathan Weese. 2013. Umbc_ebiquity-core: semantic textual similarity systems. In *Second Joint Conference on Lexical and Computational Semantics (* SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, volume 1, pages 44–52.

Zellig S Harris. 1954. Distributional structure. *Word*, 10(2-3):146–162.

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

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *KDD*.

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

Tom Kenter, Alexey Borisov, and Maarten de Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. In *ACL*.

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

Diederik P. Kingma, Tim Salimans, and Max Welling. 2016. Improving variational inference with inverse autoregressive flow. *CoRR*, abs/1606.04934.

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

Rémi Lebret and Ronan Collobert. 2014. Word embeddings through hellinger pca. In *EACL*.

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *TACL*, 3:211–225.

Jiwei Li and Eduard H. Hovy. 2014. A model of coherence based on distributed sentence representation. In *EMNLP*.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING*.

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

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 *LREC*.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2017. Advances in pre-training distributed word representations. *CoRR*, abs/1712.09405.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *NIPS*.

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

Bo Pang and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *ACL*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *ACL*.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *ICML*.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *NIPS-W*.

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

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke S. Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *CoRR*.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. In *ICML*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *EMNLP*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *ACL*.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. 2017. Rethinking skip-thought: A neighborhood based approach. In *RepL4NLP, ACL Workshop*.

Shuai Tang, Hailin Jin, Chen Fang, Zhaowen Wang, and Virginia R. de Sa. 2018. Speeding up context-based sentence representation learning with non-autoregressive convolutional decoding. In *Rep4NLP@ACL*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language Resources and Evaluation*, 39:165–210.

John Wieting, Mohit Bansal, Kevin Gimpel, and Karen Livescu. 2015. From paraphrase database to compositional paraphrase model and back. *TACL*, 3:345–358.

John Wieting and Kevin Gimpel. 2017. Revisiting recurrent networks for paraphrastic sentence embeddings. In *ACL*.

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

John Wieting and Douwe Kiela. 2019. No training required: Exploring random encoders for sentence classification. In *International Conference on Learning Representations*.

Han Zhao, Zhengdong Lu, and Pascal Poupart. 2015. Self-adaptive hierarchical sentence model. In *IJCAI*.

Yukun Zhu, Ryan Kiros, Richard S. Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. *ICCV*, pages 19–27.