The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Deconvolutional Latent-Variable Model for Text Sequence Matching

Dinghan Shen, Yizhe Zhang, Ricardo Henao, Qinliang Su, Lawrence Carin
Department of Electrical & Computer Engineering, Duke University
Durham, NC 27708
{dinghan.shen, yizhe.zhang, ricardo.henao, qinliang.su, lcarin}@duke.edu

Abstract

A latent-variable model is introduced for text matching, inferring sentence representations by jointly optimizing generative and discriminative objectives. To alleviate typical optimization challenges in latent-variable models for text, we employ deconvolutional networks as the sequence decoder (generator), providing learned latent codes with more semantic information and better generalization. Our model, trained in an unsupervised manner, yields stronger empirical predictive performance than a decoder based on Long Short-Term Memory (LSTM), with less parameters and considerably faster training. Further, we apply it to text sequence-matching problems. The proposed model significantly outperforms several strong sentence-encoding baselines, especially in the semi-supervised setting.

Introduction

The ability to infer the degree of match between two text sequences, and determine their semantic relationship, is of central importance in natural language understanding and reasoning (Bordes et al. 2014). With recent advances in deep neural networks, considerable research has focused on developing *end-to-end* deep learning models for text sequence matching (Hu et al. 2014; Wang and Jiang 2017; Rocktäschel et al. 2015; Wang, Hamza, and Florian 2017; Shen et al. 2017). State-of-the-art models typically first encode the text sequences into hidden units via a Long Short term Memory (LSTM) model or a Convolutional Neural Network (CNN), and techniques like attention mechanisms (Rocktäschel et al. 2015) or memory networks (Hill et al. 2015) are subsequently applied for the final sequence matching, usually addressed as a classification problem. However, the word-by-word matching nature of these models typically gives rise to high computational complexity, either $\mathcal{O}(T^2)$ (Wang and Jiang 2017) or $\mathcal{O}(T)$ (Rocktäschel et al. 2015), where T is the sentence length. Therefore, these approaches are computationally expensive and difficult to scale to large datasets or long text sequences.

Another class of models for matching natural language sentences is based on *sentence encoding* methods, where each sentence is mapped to a vector (embedding), and two such vectors are used for predictions of relationships between the corresponding two sentences (Bowman et al. 2016a; Mou et al. 2015). In this case the matching complexity is independent of sentence length. However, it has been found that is hard to encode the semantic information of an entire sequence into a single vector (Bowman et al. 2015).

For these models, it is important to learn an informative sentence representation with two properties: (*i*) it preserves its fundamental details, *e.g.*, n-gram fragments within the sequence of text; (*ii*) the learned representation should contain discriminative information regarding its relationship with the target sequence. So motivated, we propose to infer the embedding for each sentence with *deep generative models*, due to their ability to make effective use of unlabeled data and learn abstract features from complex data (Kingma et al. 2014; Yang et al. 2017; Pu et al. 2016; Wang et al. 2017). Moreover, the objective of a generative model addresses generation/reconstruction, and thus learns latent codes that naturally preserve essential information of a sequence, making them particularly well suited to sentence matching.

Recent advances in neural variational inference have manifested deep latent-variable models for text (Miao, Yu, and Blunsom 2016). The general idea is to map the sentence into a continuous latent variable, or *code*, via an inference network (encoder), and then use the generative network (decoder) to reconstruct the input sentence conditioned on samples from the latent code (via its posterior distribution). As a first attempt, (Bowman et al. 2016b) proposed a Variational Auto-Encoder (VAE)-based generative model for text, with LSTM networks (Hochreiter and Schmidhuber 1997) as the sequence decoder. However, due to the recurrent nature of the LSTM decoder, the model tends to largely ignore information from the latent variable; the learned sentence embedding contains little information from the input, even with several training modifications (Bowman et al. 2016b). To mitigate this issue, (Yang et al. 2017) proposed to use a dilated CNN, rather than an LSTM, as a decoder in their latent-variable model. Since this decoder is less dependent on the contextual information from previous words, the latent-variable representation tends to encode more information from the input sequence.

Unfortunately, regardless of whether LSTMs or dilated

Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

CNNs are used as the generative network, ground-truth words need to be fed into the decoder during training, which has two potential issues: (*i*) given the powerful recursive and autoregressive nature of these decoders, the latent-variable model tends to ignore the latent vector altogether, thus reducing to a *pure* language model (without external inputs) *i.e.*, latent representations are not effective during training (Bowman et al. 2016b; Chen et al. 2017); (*ii*) the learned latent vector does not necessarily encode all the information needed to reconstruct the entire sequence, since additional guidance is provided while generating every word, *i.e.*, *exposure bias* (Ranzato et al. 2016).

We propose *deconvolutional networks* as the sequence decoder in a latent-variable model, for matching natural language sentences. Without any recurrent structure in the decoder, the typical optimization issues associated with training latent-variable models for text are mitigated. Further, global sentence representations can be effectively learned, since no ground-truth words are made available to the decoder during training.

In the experiments, we first evaluate our deconvolution-based model in an unsupervised manner, and examine whether the learned embedding can automatically distinguish different writing styles. We demonstrate that the latent codes from our model are more informative than LSTM-based models, while achieving higher classification accuracy. We then apply our latent-variable model to text-sequence matching tasks, where predictions are made only based on samples from the latent variables. Consequently, without any prior knowledge on language structure, such as that used in traditional text analysis approaches (*e.g.*, via a parse tree), our deconvolutional latent-variable model outperforms several competitive baselines, especially in the semi-supervised setting.

Our main contributions are as follows:

i) We propose a neural variational inference framework for matching natural language sentences, which effectively leverages unlabeled data and achieves promising results with little supervision.

ii) We employ deconvolutional networks as the sequence decoder, alleviating the optimization difficulties of training latent-variable models for text, resulting in more informative latent sentence representations.

iii) The proposed deconvolutional latent-variable model is highly parallelizable, with less parameters and much faster training than LSTM-based alternatives.

Background

Matching natural language sentences

Assume we have two sentences for which we wish to compute the degree of match. For notational simplicity, we describe our model in the context of Recognizing Textual Entailment (RTE) (Rocktäschel et al. 2015), thus we denote the two sequences as *P* for premise and *H* for hypothesis, where each sentence pair can be represented as (p_i, h_i), for $i = 1, 2, 3..., N$, where N is the total number of pairs. The goal of sequence matching is to predict judgement y_i for the corresponding sentence pair, by modeling the conditional distribution $p(y_i|p_i, h_i)$, where $y_i \in \{$*entailment*, *contradiction*, *neutral*$\}$. *Entailment* indicates that p_i and h_i can be inferred from each other, *contradiction* suggests they have opposite semantic meanings, while *neutral* means p_i and h_i are irrelevant to each other. This framework can be generalized to other natural language processing applications, such as paraphrase identification, where $y_i = 1$ if p_i is a paraphrase of h_i, and $y_i = 0$ otherwise. In this regard, text sequence matching can be viewed as either a binary or multi-class classification problem (Yu et al. 2014).

Although word/phrase-level attention (Rocktäschel et al. 2015) or matching strategies (Wang and Jiang 2017) are often applied to text sequence-matching problems, we only consider *sentence encoding-based models*, because of their promising low complexity. Specifically, our model is based on the *siamese* architecture (Bromley et al. 1994), which consists of a twin network that processes natural language sentence pairs independently (the parameters of the twin network are tied); there is no interaction before both sentence representations are inferred. A classification layer is built on top of the two latent representations, for final prediction (matching).

The shared encoder network can be designed as any form of nonlinear transformation, including Convolutional Neural Networks (CNNs), Recurrent Neural Networks (RNNs) or Multi-Layer Perceptrons (MLPs). However, to effectively match natural language sentences with the *siamese* architecture, the key is to learn informative sentence representations through the encoder network. To this end, below we describe use of CNNs in the context of a latent-variable model.

Latent-variable models for text processing

Sequence-to-sequence models (Sutskever, Vinyals, and Le 2014) are the most common strategy for obtaining robust sentence representations, as these are capable of leveraging information from unlabeled data. These models first encode the input sentence x (composed of T words, $w_{1:T}$) into a fixed-length vector $z = g(x)$, and then reconstruct/generate the output sequence from z. Specifically, in the autoencoder setup, the output of the decoder is the reconstruction of the input sentence x, denoted $\hat{x}$ with words $\hat{w}_{1:T}$,

$$\begin{aligned} p(\hat{x}|x) &= p(\hat{w}_{1:T}|w_{1:N}) \\ &= p(\hat{w}_1|z = g(x)) \prod_{t=2}^{T} p(\hat{w}_t|z = g(x), \hat{w}_{1:t-1}) \,, \end{aligned} \tag{1}$$

where $g(\cdot)$ is a *deterministic*, generally nonlinear transformation of x. The deterministic $g(x)$ may result in poor model generalization, especially when only a limited number of labeled data are available for training. Below we consider a *probabilistic* representation for z, *i.e.,* $p(z|x)$.

Recently (Miao, Yu, and Blunsom 2016) introduced a Neural Variational Inference (NVI) framework for text modeling, in which they infer a stochastic latent variable $z \sim q(z|x)$ to model the input text, constructing an inference network to approximate the true posterior distribution $p(z|x)$. This strategy endows latent variable z with a better ability to generalize (Miao, Yu, and Blunsom 2016). Conditioning

on the latent code z, a decoder network $p(x|z)$ maps z back to reconstruct the original sequence, x. Given a set of observed sentences (training set), the parameters of this model are learned by maximizing the marginal $p(x)$. Since this is intractable in most cases, a variational lower bound is typically employed as the objective to be maximized (Kingma and Welling 2013):

$$\begin{aligned}\mathcal{L}_{\text{vae}} &= \mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)] - D_{KL}(q_\phi(z|x)|p(z)) \\ &= E_{q_\phi(z|x)}[\log p_\theta(x|z) + \log p(z) - \log q_\phi(z|x)] \\ &\leq \log \int p_\theta(x|z)p(z)dz = \log p_\theta(x)\,, \end{aligned} \tag{2}$$

where θ and ϕ denote decoder and encoder parameters, respectively. The lower bound $\mathcal{L}_{\text{vae}}(\theta, \phi; x)$ is maximized w.r.t. both encoder and decoder parameters. Intuitively, the model aims to minimize the reconstruction error as well as to regularize the posterior distribution $q_\phi(z|x)$ as to not diverge too much from the prior $p(z)$. This neural variational inference framework has achieved significant success on other types of data, such as images (Gregor et al. 2015; Pu et al. 2016).

Challenges with the NVI framework for text

Extracting sentence features for text with the above NVI framework has been shown to be difficult (Bowman et al. 2016b; Yang et al. 2017). For an unsupervised latent-variable model, which is often referred to as a variational autoencoder (Kingma and Welling 2013), the parameters are optimized by minimizing the reconstruction error of sentences, as well as regularizing the posterior distribution $q_\phi(z|x)$ to be close to the prior $p(z)$, as in (2) via $D_{KL}(q_\phi(z|x)|p(z))$. Therefore, we can think of the variational autoencoder as a regularized version of a standard (deterministic) autoencoder (sequence-to-sequence model), due to the additional penalty term coming from KL divergence loss.

Although the KL divergence in (2) term plays a key role in training latent-variable models with the NVI framework, it has been reported that, when applied to text data (sentences), the KL loss tends to be insignificantly small during training (Bowman et al. 2016b). As a result, the encoder matches the Gaussian prior regardless of the input, and the decoder doesn't take advantage of information from the latent variable z. Moreover, it has been reported that poor results in this setting may be attributed to the autoregressive nature of the LSTM decoder (Chen et al. 2017; Bowman et al. 2016b). While decoding, the LSTM imposes strong conditional dependencies between consecutive words, thus, from (1), the information from z becomes less impactful during learning. Motivated by these issues, (Yang et al. 2017) employed dilated CNNs, instead of the LSTM, as a sentence decoder for a latent-variable model. In (Yang et al. 2017) the latent variable z is able to encode more semantic information, because of the smaller contextual capacity of the dilated CNN decoder. However, optimization challenges remain, because ground-truth words are employed while training, as the dilated CNN is an autoregressive decoder. Consequently, the inferred latent codes cannot be considered as global features of a sentence, since they do not necessarily encode all the information needed to reconstruct an entire sequence.

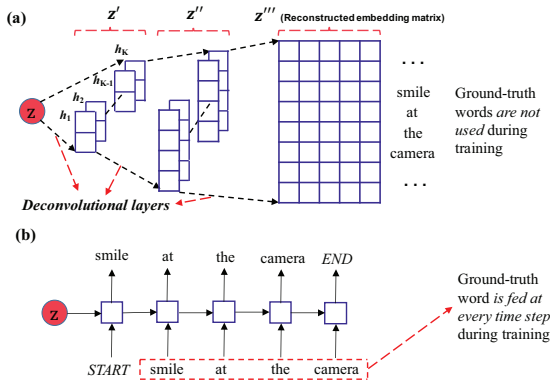

Figure 1: (a) Diagram of deconvolutional sequence decoder, comparing with **(b)** LSTM sequence decoder. Notably, in contrast to a LSTM decoder, ground truth words are not provided for the deconvolutional networks during training. As a result, the failure mode of optimization described in (Bowman et al. 2016b), where the KL divergence term is vanishingly small, is largely mitigated.

Model

Deconvolutional sequence decoder

Deconvolutional networks, also known as *transposed* convolutional layers, are typically used in deep learning models to up-sample fixed-length latent representations or high-level feature maps (Zeiler et al. 2010). Although widely adopted in image generative models, deconvolutional networks have been rarely applied to generative models for text. To understand the form of the decoder needed for text, we first consider the associated convolutional encoder ((Kim 2014), (Zhang et al. 2017b)). The text is represented as a matrix, with "width" dictated by the sentence length and "height" dictated by the dimensionality of the word embeddings. With K_1 convolutional filters at layer 1 of the model, after one-dimensional (1D) convolution between the 2D filters and 2D sentence embedding matrix (convolution in the direction of the word index, or "time"), K_1 1D signals are manifested. Using these K_1 1D feature maps, a similar process repeats to substantiate subsequent layers in the deep model. Hence, at layer l of the model, there are K_l 1D signals manifested from K_l 1D convolutions between K_l 2D filters and the 2D feature-map from layer $l-1$.

The encoder discussed above starts at the "bottom" with the sentence-embedding matrix, and works upward to the latent code z. The decoder works downward, starting at z and arriving at the sentence-embedding matrix. Specifically, the decoder network takes as input $z \in \mathbb{R}^M$ sampled from the inference (encoder) network $q_\phi(z|x)$. For an L-layer decoder model, the feature maps at layer L (just beneath the latent code z) are manifested by K_L filter matrices

$f_i^{(L)} \in \mathbb{R}^{H_L \times M}$, for $i = 1, 2, ..., K_L$, where H_L corresponds to the number of components in the temporal (word) dimension. Each 2D matrix $f_i^{(L)}$ is multiplied by column vector z (transpose convolution), yielding K_L 1D feature maps. This yields an $H_L \times K_L$ feature-map matrix at layer L (followed by ReLU pointwise nonlinearity). To yield the layer $L-1$ feature map matrix, the process repeats, using filters $f_i^{(L-1)} \in \mathbb{R}^{H_{L-1} \times K_L}$, for $i = 1, 2, ..., K_{L-1}$, with which K_{L-1} 1D convolutions are performed with the feature-map matrix from layer L (convolutions in the temporal/word dimension). This again yields a feature-map matrix at layer $L-1$, followed by ReLU nonlinearity.

This process continues sequentially, until we arrive at the bottom of the decoder network, yielding a final matrix from which the sentence-embedding matrix is approximated. To be explicit, in Fig. 1 let z' and z'' represent the feature-map matrices at the top-two layers of a three-layer model. Let z''' represent the matrix recovered at the bottom layer of the network through the above process, with "height" corresponding to the dimension of the word-embedding. Suppose $\boldsymbol{E}$ is the word-embedding matrix for our vocabulary, and $\hat{w}_i$ the ith word in the reconstructed sentence. We compute the probability that $\hat{w}_i$ is word s as:

$$p(\hat{w}_i = s) = \frac{\exp\{\tau^{-1} \cos(z_i''', \boldsymbol{E}[s])\}}{\sum_{s' \in V} \exp\{\tau^{-1} \cos(z_i''', \boldsymbol{E}[s'])\}}, \quad (3)$$

where $\cos(a, b)$ is the *cosine similarity* between vectors a and b, V is the vocabulary which contains all possible words and $\boldsymbol{E}[s]$ represents the column of $\boldsymbol{E}$ corresponding to word s; z_i''' is the i-th column of the up-sampled representation z'''. Parameter τ controls the sparsity of resulting probabilities, which we denote as the *temperature* parameter. We set $\tau = 0.01$ in our experiments.

The multilayer coarse-to-fine process (latent variable vector to embedding matrix) implied by repeatedly applying the above decoder process illustrated in Figure 1(a)) has two advantages: i) it reflects the natural hierarchical tree structure of sentences, thus may better represent syntactic features, which is useful when reconstructing sentences; ii) the deconvolutional network allows for efficient parallelization while generating each fragment of a sentence, and thus can be considerably faster than an LSTM decoder.

As shown in Figure 1, the training procedures for deconvolutional (a) and LSTM (b) decoders are intrinsically different. In the latter, ground-truth words of the previous time steps are provided while training the network. In contrast, the deconvolutional network generates the entire sentence (in block) from z alone. Because of this distinction, the LSTM decoder, as an autoregressive model with powerful recurrence, tends to explain all structure in the data, with little insight from the latent variables which only provide information at the beginning of the sentence, thus acting merely as a prior.

Deconvolutional latent-variable models

In this section we incorporate the deconvolutional sequence decoder described in the previous section in our latent-variable model for text. Because of the coarse-to-fine generation process described above, the model does not have partial access to observed data (ground-truth words) during the generation process, as in an LSTM, thus the latent-variable model must learn to encode as much information as possible from the input alone. Moreover, in this way the learned latent code can be truly viewed as a global feature representation of sentences, since it contains all the essential information to generate the text sequence. In the following, we describe the proposed deconvolutional latent-variable models, in the context of both unsupervised and supervised (including semi-supervised) learning.

Unsupervised sequence learning To demonstrate the effectiveness of our proposed model, we explore training it in an unsupervised manner. Specifically, for a input sentence x, the latent code is inferred through an encoder network $q_\phi(z|x)$ implemented as

$$\mu = g_1(f^{\text{cnn}}(x; \phi_{10}); \phi_{11}), \quad \log \sigma = g_2(f^{\text{cnn}}(x; \phi_{20}); \phi_{21})$$
$$\varepsilon \sim \mathcal{N}(0, \mathbf{I}), \qquad z = \mu + \varepsilon \odot \sigma, \quad (4)$$

where $f^{\text{cnn}}(x; \phi_{10})$ denotes the transformation function of the encoder, accomplished via learning a CNN with input x and parameters ϕ_{10}, and $\odot$ represents the Hadamard vector product. The posterior mean μ and variance σ are generated through two non-linear transformations $g_1(\cdot)$ and $g_2(\cdot)$, both parameterized as neural networks; $g_1(y; \phi_{11})$ has input y and parameters ϕ_{11}. Note that (4) is $q_\phi(z|x)$ in (2), where $\phi = \{\phi_{10}, \phi_{11}, \phi_{20}, \phi_{21}\}$. Then z is sampled with the reparameterization trick (Kingma and Welling 2013) to facilitate model training. The sampled z is then fed into a deconvolutional sequence decoder described above, to reconstruct the corresponding input sentences. The model is trained by optimizing the variational lower bound in (2), without any discriminative information.

Supervised sequence matching We apply our latent-variable model to text sequence-matching problems, employing the discriminative information encoded in latent code z (see Figure 2). For a sentence pair (p_i, h_i), the latent code for each sequence is inferred as in (4), where the parameters of the encoder network for z_p and z_h, premise and hypothesis, respectively, are shared. They are decoded by two shared-weight deconvolution networks, to recover the corresponding input sentence.

To infer the label, y, the two latent features are again sampled from the inference network and processed by a matching layer, to combine the information in the two sentences. This matching layer, defined as *heuristic* matching layer by (Mou et al. 2015), can be specified as:

$$m = [z_p; z_h; z_p - z_h; z_p \odot z_h],$$

These matching features are stacked together into $m \in \mathbb{R}^{4M}$, for $z_p, z_h \in \mathbb{R}^M$, and fed into a classifier. The classifier is a two-layer MLP followed by a fully-connected softmax layer, that outputs the probabilities for each label (entailment, contradiction and neutral), to model the conditional distribution $p_\psi(y|z_p, z_h)$, with parameters ψ.

To allow the model to explore and balance between maximizing the variational lower bound and minimizing the se-

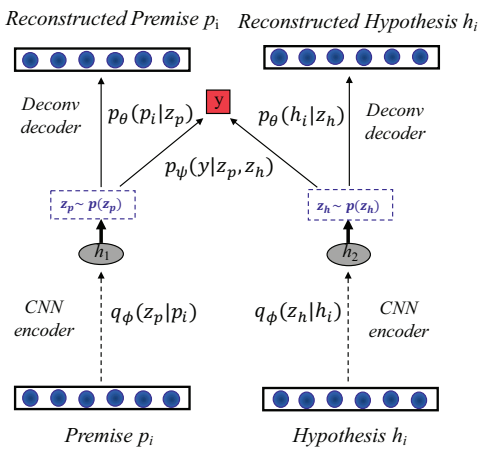

Figure 2: Our deconvolutional latent-variable model for text sequence matching. The reconstruction/generation and discriminative objectives are jointly optimized to learn more robust latent codes for sentences.

quence matching loss, a joint training objective is employed:

$$\mathcal{L}^{\text{label}} = -\mathcal{L}_{\text{vae}}(\theta, \phi; p_i) - \mathcal{L}_{\text{vae}}(\theta, \phi; h_i) + \alpha \mathcal{L}_{\text{match}}(\psi; z_p, z_h, y)\,,$$

where ψ refers to parameters of the MLP classifier and α controls the relative weight between the generative loss, $\mathcal{L}_{\text{vae}}(\cdot)$, and sequence matching loss, $\mathcal{L}_{\text{match}}(\cdot)$, defined as the cross-entropy loss. When implementing this model, we anneal the value of α during training from 0 to 1 (the annealing rate is treated as a hyperparameter), so that the latent variable learned can gradually focus less on the reconstruction objective, only retaining those features that are useful for sequence matching, *i.e.*, minimizing the second term.

Extension to semi-supervised learning Our latent-variable model can be readily extended to a semi-supervised scenario, where only a subset of sequence pairs have corresponding class labels. Suppose the empirical distributions for the labeled and unlabeled data are referred to as $\tilde{p}_l(P, H, y)$ and $\tilde{p}_u(P, H)$, respectively. The loss function for unlabeled data can be expressed as:

$$\mathcal{L}^{\text{unlabel}} = -\mathcal{L}_{\text{vae}}(\theta, \phi; p_i) - \mathcal{L}_{\text{vae}}(\theta, \phi; h_i)\,.$$

Therefore, the overall objective for the joint latent-variable model is:

$$\mathcal{L}_{\text{joint}} = \mathbb{E}_{(p_i, h_i, y) \sim \tilde{p}_l}[\mathcal{L}^{\text{label}}(p_i, h_i, y)] + \mathbb{E}_{(p_i, h_i) \sim \tilde{p}_u}[\mathcal{L}^{\text{unlabel}}(p_i, h_i)]\,. \quad (5)$$

To minimize $\mathcal{L}_{\text{joint}}$ w.r.t. θ, ϕ and ψ, we employ Monte Carlo integration to approximate the expectations in (5). In this case unlabeled data are leveraged in the objective via the standard VAE lower bound. During training, all parameters are jointly updated with stochastic gradient descent (SGD).

Dataset	Train	Test	Classes	Vocabulary
Quora	384348	10000	2	10k
SNLI	549367	9824	3	20k

Table 1: Summary of text sequence matching datasets.

Experiments

Experimental Setup

Our deconvolutional latent-variable model can be trained in an unsupervised, supervised or semi-supervised manner. In this section we first train the model in an unsupervised way, with a mixed corpus of scientific and informal writing styles, and evaluate the sentence embeddings by checking whether they can automatically distinguish different sentence characteristics, *i.e.*, writing styles. Further, we apply our models to two standard text sequence matching tasks: Recognizing Textual Entailment (RTE) and paraphrase identification, in a semi-supervised setting. The summary statistics of both datasets are presented in Table 1.

For simplicity, we denote our deconvolutional latent-variable model as DeConv-LVM in all experiments. To facilitate comparison with prior work, several baseline models are implemented: (*i*) a basic Siamese model with CNNs as the encoder for both sentences, with sharing configurations and weights; (*ii*) an auto-encoder with CNN as the sequence encoder and DeConv as decoder; 3) a latent-variable model using a CNN as the inference network, and the generative network is implemented as an LSTM (denoted LSTM-LVM).

We use 3-layer convolutional neural networks for the inference/encoder network, in order to extract hierarchical representation of sentences ((Hu et al. 2014)). Specifically, for all layers we set the filter window size (W) as 5, with a stride of 2. The feature maps (K) are set as 300, 600, 500, for layers 1 through 3, respectively. In our latent-variable models, the 500-dimension feature vector is then fed into two MLPs to infer the mean and variance of the latent variable z. The generative/decoder network is implemented as 3-layer deconvolutional networks, to decode the samples from latent variable z of size $M = 500$.

The model is trained using Adam (Kingma and Ba 2014) with a learning rate of 3×10^{-4} for all parameters. Dropout (Srivastava et al. 2014) is employed on both word embedding and latent variable layers, with rates selected from $\{0.3, 0.5, 0.8\}$ on the validation set. We set the mini-batch size to 32. In semi-supervised sequence matching experiments, L_2 norm of the weight vectors is employed as a regularization term in the loss function, and the coefficient of the L_2 loss is treated as a hyperparameter and tuned on the validation set. All experiments are implemented in Tensorflow (Abadi et al. 2016), using one NVIDIA GeForce GTX TITAN X GPU with 12GB memory.

Unsupervised Sentence Embedding

To investigate the effectiveness of our latent-variable model, we first train it in an unsupervised manner, using the dataset in (Zhang et al. 2017a), where sentences from two corpora,

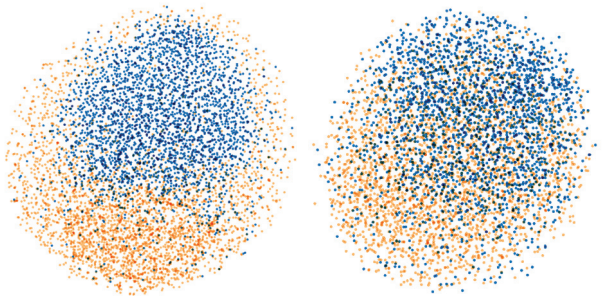

Figure 3: t-SNE embeddings of latent codes (**left**: DeConv-LVM, **right**: LSTM-LVM) for *BookCorpus* and *arXiv* sentences, which are colored as orange and blue, respectively.

i.e, *BookCorpus* dataset (Zhu et al. 2015) and the *arXiv* dataset, are merged together in equal proportion. The motivation here is to check whether the latent codes learned in our model can automatically distinguish between different writing styles, *i.e.*, sentences with scientific or informal styles represented by *BookCorpus* and *arXiv* dataset, respectively. In this experiment, our model is trained by optimizing the variational lower bound in (2), without any label/discriminative information provided. We compare our model with another latent-variable model using LSTM as the decoder, to especially highlight the contribution of the deconvolutional network to the overall setup. To ensure a fair comparison, we employ the same model architecture for the LSTM-based latent-variable model (LSTM-LVM), except for the decoder utilized. The LSTM hidden-state dimension is set to 500, with the latent variable z fed to decoder as input at every time step.

After the models converge, we randomly sample 5000 sentences from the test set and map their 500-dimensional latent embeddings, z, to a 2D vector using t-SNE (Maaten and Hinton 2008). The embedding plots for DeConv-LVM (left) and LSTM-LVM (right) are shown in Figure 3. For both cases, the plot shape of sampled latent embeddings is very close to a circle, which means the posterior distribution $p(z|x)$ matches the Gaussian prior $p(z)$ well. More importantly, when we use deconvolutional networks as the decoder, disentangled latent codes for the two writing styles can be clearly observed in the majority of prior space. This indicates that the semantic meanings of a sentence are encoded into the latent variable z, even when we train the model in an unsupervised manner. On the contrary, the latent codes of LSTM-LVM inferred for different writing styles tend to mix with each other, and cannot be separated as easily as in the case of Deconv-LVM, suggesting that less information may be encoded into the embeddings.

To better understand the advantages of deconvolutional networks as the decoder in the latent-variable models, we perform a quantitative comparison between the latent codes in DeConv-LVM and LSTM-LVM. In Table 2 we show the number of parameters, training time for 10,000 iterations, and the percentage of KL loss in the total loss for both models. Moreover, we extract sentence features from each

Model	# params	Time	KL	Acc
LSTM-LVM	$\sim$ 16 million	39m 41s	4.6	91.7
DeConv-LVM	$\sim$ 12 million	**8m 23s**	31.7	**96.2**

Table 2: Quantitative comparison between latent-variable models with LSTM and deconvolutional networks as the sentence decoder.

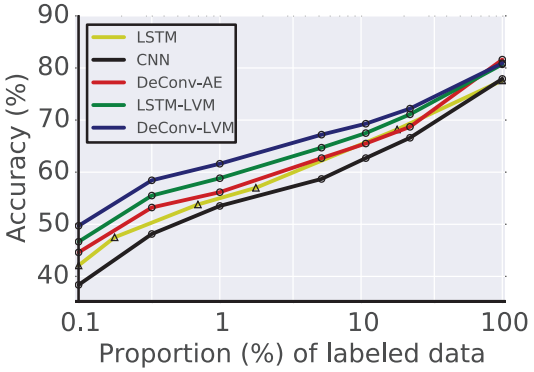

Figure 4: The performance of various models on SNLI dataset, with different amount of labeled data.

model, and train a linear classifier on top, to distinguish between scientific and informal writing styles. The sentence embeddings are fixed during training, in order to elucidate the quality of latent codes learned in an unsupervised manner. 1000 sentences are sampled from the training set to learn the classifier and the classification accuracy is calculated on the whole test set. DeConv-LVM (96.2%) performs better than LSTM-LVM (91.7%), again indicating that the the latent codes of DeConv-LVM are more informative. This observation corresponds well with the fact that the percentage of KL loss in DeConv-LVM (31.7%) is much larger than in LSTM-LVM (4.6%), where larger KL divergence loss can be considered as a sign that more useful information has been encoded in the latent variable z (Bowman et al. 2016b; Yang et al. 2017). Further, we observe that DeConv-LVM has relatively few parameters compared to LSTM-LVM, making it a promising latent-variable model for text.

Recognizing Textual Entailment (RTE)

Motivated by the superior performance of our deconvolutional latent-variable model on unsupervised learning, we further apply it to text sequence matching, in a semi-supervised scenario. We consider the task of recognizing text entailment on the Stanford Natural Language Inference (SNLI) dataset (Bowman et al. 2015).

To check the generalization ability of our latent variable learned, we experimented with different amounts of labeled training data (other sentence pairs in the training set are used as unlabeled data). The results are shown in Figure 4. Compared to the LSTM baseline models in (Bowman et al. 2015) and our basic CNN implementation, both our autoencoder and latent-variable models make use of the unlabeled data and achieve better results than simply train an

Model	28k	59k	120k
LSTM ((Kim et al. 2017))	57.9	62.5	65.9
LSTM-AE ((Kim et al. 2017))	59.9	64.6	68.5
LSTM-ADAE ((Kim et al. 2017))	62.5	66.8	70.9
CNN (random)	58.7	62.7	65.6
CNN (Glove)	60.3	64.1	66.8
DeConv-AE	62.1	65.5	68.7
LSTM-LVM	64.7	67.5	71.1
DeConv-LVM	**67.2**	**69.3**	**72.2**

Table 3: Semi-supervised recognizing textual entailment accuracy on SNLI dataset, in percentage. For direct comparison with (Kim et al. 2017). The number of labeled examples is set as 28k, 59k or 120k.

encoder network, *i.e.*, LSTM, CNN, only with the labeled data. More importantly, the DeConv-LVM we propose outperforms LSTM-LVM in all cases, consistent with previous observations that the latent variable z in our DeConv-LVM tends to be more informative. Note that when using all labeled data when training, DeConv-AE (81.6%) performs a bit better than DeConv-LVM (80.9%), which is not surprising since DeConv-LVM introduces a further constraint on the latent features learned (close to prior distribution) and may not be optimal when a lot of labeled data are available for training.

To directly compare with (Kim et al. 2017) on semi-supervised learning experiments, we follow their experiment setup where 28k, 59k, 120k labeled examples are used for training. According to Table 3, it turns out that our DeConv-AE model is a competitive baseline, and outperform their LSTM-AE results. Moreover, our DeConv-LVM achieves even better results than DeConv-AE and LSTM-LVM, suggesting that the deconvolution-based latent-variable model we propose makes effective use of unsupervised information. Further, we see that the gap tends to be larger when the number of labeled data is smaller, further demonstrating that DeConv-LVM is a promising strategy to extract useful information from unlabeled data.

Paraphrase Identification

We investigate our deconvolutional latent-variable model on the paraphrase identification task with the Quora Question Pairs dataset, following the same dataset split as (Wang, Hamza, and Florian 2017). We consider cases where 1k, 5k, 10k, 25k labeled examples are used for training. As illustrated in Table 4, a CNN encoder with Glove pre-trained word embeddings consistently outperforms that with randomly initialized word embeddings, while the autoencoder model achieves better results than only training a CNN encoder, corresponding with findings in (Dai and Le 2015).

More importantly, our latent-variable models show even higher accuracy than autoencoder models, demonstrating that they effectively utilize the information of unlabeled data and that they represent an effective strategy for paraphrase identification task. Our DeConv-LVM again performs better than LSTM-LVM in all cases, indicating that the deconvolutional decoder can leverage more benefits from the latent-variable model. However, we can also see the trend that with larger number of labeled data, the gaps between these models are smaller. This may be attributed to the fact that when lots of labeled data are available, discriminative information tends be the dominant factor for better performance, while the information from unlabeled data becomes less important.

Model	1k	5k	10k	25k
CNN (random)	56.3	59.2	63.8	68.9
CNN (Glove)	58.5	62.4	66.1	70.2
LSTM-AE	59.3	63.8	67.2	70.9
DeConv-AE	60.2	65.1	67.7	71.6
LSTM-LVM	62.9	67.6	69.0	72.4
DeConv-LVM	**65.1**	**69.4**	**70.5**	**73.7**

Table 4: Paraphrase identification accuracy on Quora Question Pairs dataset, in percentages.

Related Work

The proposed framework is closely related to recent research on incorporating NVI into text modeling (Bowman et al. 2016b; Miao, Yu, and Blunsom 2016; Xu et al. 2017; Zhang et al. 2016; Serban et al. 2017). (Bowman et al. 2016b) presented the first attempt to utilize NVI for language modeling, but their results using an LSTM decoder were largely negative. (Miao, Yu, and Blunsom 2016) applied the NVI framework to an unsupervised bags-of-words model. However, from the perspective of text representation learning, their model ignores word-order information, which may be suboptimal for downstream supervised tasks. (Xu et al. 2017) employed a variational autoencoder with the LSTM-LSTM architecture for semi-supervised sentence classification. However, as illustrated in our experiments, as well as in (Yang et al. 2017), the LSTM decoder is not the most effective choice for learning informative and discriminative sentence embeddings.

The NVI framework has also been employed for text-generation problems, such as machine translation (Zhang et al. 2016) and dialogue generation (Serban et al. 2017), with the motivation to improve the diversity and controllability of generated sentences. Our work is distinguished from this prior research in two principal respects: (*i*) We leveraged the NVI framework for latent variable models to text sequence matching tasks, due to its ability to take advantage of unlabeled data and learn robust sentence embeddings; (*ii*) we employed deconvolutional networks, instead of the LSTM, as the decoder (generative) network. We demonstrated the effectiveness of our framework in both unsupervised and supervised (including semi-supervised) learning cases.

Conclusion

We have presented a latent variable model for matching natural language sentences, with deconvolutional networks as the sequence encoder. We show that by jointly optimizing the variational lower bound and matching loss, the model is effective at inferring robust sentence representations for determining their semantic relationship, even with limited

amount of labeled data. State-of-the-art experimental results on two semi-supervised sequence matching tasks are achieved, demonstrating the advantages of our approach. This work provides a promising strategy towards training effective and fast latent-variable models for text data.

Acknowledgements This research was supported in part by ARO, DARPA, DOE, NGA and ONR.

References

Abadi, M.; Agarwal, A.; Barham, P.; Brevdo, E.; Chen, Z.; Citro, C.; Corrado, G. S.; Davis, A.; Dean, J.; Devin, M.; et al. 2016. Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Bordes, A.; Glorot, X.; Weston, J.; and Bengio, Y. 2014. A semantic matching energy function for learning with multi-relational data. *Machine Learning* 94(2):233–259.

Bowman, S. R.; Angeli, G.; Potts, C.; and Manning, C. D. 2015. A large annotated corpus for learning natural language inference. *EMNLP*.

Bowman, S. R.; Gauthier, J.; Rastogi, A.; Gupta, R.; Manning, C. D.; and Potts, C. 2016a. A fast unified model for parsing and sentence understanding. *ACL*.

Bowman, S. R.; Vilnis, L.; Vinyals, O.; Dai, A. M.; Jozefowicz, R.; and Bengio, S. 2016b. Generating sentences from a continuous space. *CoNLL* 10.

Bromley, J.; Guyon, I.; LeCun, Y.; Säckinger, E.; and Shah, R. 1994. Signature verification using a" siamese" time delay neural network. In *NIPS*, 737–744.

Chen, X.; Kingma, D. P.; Salimans, T.; Duan, Y.; Dhariwal, P.; Schulman, J.; Sutskever, I.; and Abbeel, P. 2017. Variational lossy autoencoder. *ICLR*.

Dai, A. M., and Le, Q. V. 2015. Semi-supervised sequence learning. In *NIPS*, 3079–3087.

Gregor, K.; Danihelka, I.; Graves, A.; Rezende, D. J.; and Wierstra, D. 2015. Draw: A recurrent neural network for image generation. *ICML*.

Hill, F.; Bordes, A.; Chopra, S.; and Weston, J. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv preprint arXiv:1511.02301*.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Hu, B.; Lu, Z.; Li, H.; and Chen, Q. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*, 2042–2050.

Kim, Y.; Zhang, K.; Rush, A. M.; LeCun, Y.; et al. 2017. Adversarially regularized autoencoders for generating discrete structures. *arXiv preprint arXiv:1706.04223*.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *EMNLP*.

Kingma, D., and Ba, J. 2014. Adam: A method for stochastic optimization. *ICLR*.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kingma, D. P.; Mohamed, S.; Rezende, D. J.; and Welling, M. 2014. Semi-supervised learning with deep generative models. In *NIPS*, 3581–3589.

Maaten, L. v. d., and Hinton, G. 2008. Visualizing data using t-sne. *JMLR* 9(Nov):2579–2605.

Miao, Y.; Yu, L.; and Blunsom, P. 2016. Neural variational inference for text processing. In *ICML*, 1727–1736.

Mou, L.; Men, R.; Li, G.; Xu, Y.; Zhang, L.; Yan, R.; and Jin, Z. 2015. Natural language inference by tree-based convolution and heuristic matching. *ACL*.

Pu, Y.; Gan, Z.; Henao, R.; Yuan, X.; Li, C.; Stevens, A.; and Carin, L. 2016. Variational autoencoder for deep learning of images, labels and captions. In *NIPS*, 2352–2360.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2016. Sequence level training with recurrent neural networks. *ICLR*.

Rocktäschel, T.; Grefenstette, E.; Hermann, K. M.; Kočiskỳ, T.; and Blunsom, P. 2015. Reasoning about entailment with neural attention. *ICLR*.

Serban, I. V.; Sordoni, A.; Lowe, R.; Charlin, L.; Pineau, J.; Courville, A. C.; and Bengio, Y. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, 3295–3301.

Shen, D.; Min, M. R.; Li, Y.; and Carin, L. 2017. Adaptive convolutional filter generation for natural language understanding. *arXiv preprint arXiv:1709.08294*.

Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; and Salakhutdinov, R. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*, 3104–3112.

Wang, S., and Jiang, J. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Wang, W.; Pu, Y.; Verma, V. K.; Fan, K.; Zhang, Y.; Chen, C.; Rai, P.; and Carin, L. 2017. Zero-shot learning via class-conditioned deep generative models. *arXiv preprint arXiv:1711.05820*.

Wang, Z.; Hamza, W.; and Florian, R. 2017. Bilateral Multi-Perspective Matching for Natural Language Sentences. *CoRR*.

Xu, W.; Sun, H.; Deng, C.; and Tan, Y. 2017. Variational autoencoder for semi-supervised text classification. In *AAAI*, 3358–3364.

Yang, Z.; Hu, Z.; Salakhutdinov, R.; and Berg-Kirkpatrick, T. 2017. Improved variational autoencoders for text modeling using dilated convolutions. *ICML*.

Yu, L.; Hermann, K. M.; Blunsom, P.; and Pulman, S. 2014. Deep learning for answer sentence selection. *arXiv preprint arXiv:1412.1632*.

Zeiler, M. D.; Krishnan, D.; Taylor, G. W.; and Fergus, R. 2010. Deconvolutional networks. In *CVPR*, 2528–2535. IEEE.

Zhang, B.; Xiong, D.; Su, J.; Duan, H.; and Zhang, M. 2016. Variational neural machine translation. *arXiv preprint arXiv:1605.07869*.

Zhang; Gan, Z.; Fan, K.; Chen, Z.; Henao, R.; Shen, D.; and Carin, L. 2017a. Adversarial feature matching for text generation. *ICML*.

Zhang, Y.; Shen, D.; Wang, G.; Gan, Z.; Henao, R.; and Carin, L. 2017b. Deconvolutional paragraph representation learning. *NIPS*.

Zhu, Y.; Kiros, R.; Zemel, R.; Salakhutdinov, R.; Urtasun, R.; Torralba, A.; and Fidler, S. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *ICCV*, 19–27.