Generating Sentences from a Continuous Space

Samuel R. Bowman*
NLP Group and Dept. of Linguistics
Stanford University
sbowman@stanford.edu

Luke Vilnis*
CICS
UMass Amherst
luke@cs.umass.edu

Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz & Samy Bengio
Google Brain
Google, Inc.
{vinyals, adai, bengio}@google.com, rafjoz@gmail.com

Abstract

The standard recurrent neural network language model (RNNLM) generates sentences one word at a time and does not work from an explicit global sentence representation. In this work, we introduce and study an RNN-based variational autoencoder generative model that incorporates distributed latent representations of entire sentences. This factorization allows it to explicitly model holistic properties of sentences such as style, topic, and high-level syntactic features. Samples from the prior over these sentence representations remarkably produce diverse and well-formed sentences through simple deterministic decoding. By examining paths through this latent space, we are able to generate coherent novel sentences that interpolate between known sentences. We present techniques for solving the difficult learning problem presented by this model, demonstrate its effectiveness in imputing missing words, explore many interesting properties of the model's latent sentence space, and present negative results on the use of the model in language modeling.

1 Introduction

Recurrent neural network language models (RNNLMs, Mikolov et al., 2011) represent the state of the art in unsupervised generative modeling for natural language sentences. In supervised settings, RNNLM decoders conditioned on task-specific features are the state of the art in tasks like machine translation (Sutskever et al., 2014; Bahdanau et al., 2015) and image captioning (Vinyals et al., 2015; Mao et al., 2015; Donahue et al., 2015). The RNNLM generates sentences word-by-word based on an evolving distributed state representation, which makes it a probabilistic model with no significant independence assumptions, and makes it capable of modeling complex distributions over sequences, including those with long-term dependencies. However, by breaking the model structure down into a series of next-step predictions, the RNNLM does not expose an interpretable representation of global features like topic or of high-level syntactic properties.

i went to the store to buy some groceries .
i store to buy some groceries .
i were to buy any groceries .
horses are to buy any groceries .
horses are to buy any animal .
horses the favorite any animal .
horses the favorite favorite animal .
horses are my favorite animal .

Table 1: Sentences produced by greedily decoding from points between two sentence encodings with a conventional autoencoder. The intermediate sentences are not plausible English.

We propose an extension of the RNNLM that is designed to explicitly capture such global features in a continuous latent variable. Naively, maximum likelihood learning in such a model presents an intractable inference problem. Drawing inspiration from recent successes in modeling images (Gregor et al., 2015), handwriting, and natural speech (Chung et al., 2015), our model circumvents these difficulties using the architecture of a *variational autoencoder* and takes advantage of recent advances in variational inference (Kingma and Welling, 2015; Rezende et al., 2014) that introduce a practical training technique for powerful neural network generative models with latent variables.

Our contributions are as follows: We propose a variational autoencoder architecture for text and discuss some of the obstacles to training it as well as our proposed solutions. We find that on a standard language modeling evaluation where a global variable is not explicitly needed, this model yields similar performance to existing RNNLMs. We also evaluate our model using a larger corpus on the task of imputing missing words. For this task, we introduce a novel evaluation strategy using an

*First two authors contributed equally. Work was done when all authors were at Google, Inc.

adversarial classifier, sidestepping the issue of intractable likelihood computations by drawing inspiration from work on non-parametric two-sample tests and adversarial training. In this setting, our model's global latent variable allows it to do well where simpler models fail. We finally introduce several qualitative techniques for analyzing the ability of our model to learn high level features of sentences. We find that they can produce diverse, coherent sentences through purely deterministic decoding and that they can interpolate smoothly between sentences.

2 Background

2.1 Unsupervised sentence encoding

A standard RNN language model predicts each word of a sentence conditioned on the previous word and an evolving hidden state. While effective, it does not learn a vector representation of the full sentence. In order to incorporate a continuous latent sentence representation, we first need a method to map between sentences and distributed representations that can be trained in an unsupervised setting. While no strong generative model is available for this problem, three non-generative techniques have shown promise: sequence autoencoders, skip-thought, and paragraph vector.

Sequence autoencoders have seen some success in pre-training sequence models for supervised downstream tasks (Dai and Le, 2015) and in generating complete documents (Li et al., 2015a). An autoencoder consists of an encoder function φ_{enc} and a probabilistic decoder model $p(x|\vec{z} = \varphi_{enc}(x))$, and maximizes the likelihood of an example x conditioned on $\vec{z}$, the learned code for x. In the case of a sequence autoencoder, both encoder and decoder are RNNs and examples are token sequences.

Standard autoencoders are not effective at extracting for global semantic features. In Table 1, we present the results of computing a path or *homotopy* between the encodings for two sentences and decoding each intermediate code. The intermediate sentences are generally ungrammatical and do not transition smoothly from one to the other. This suggests that these models do not generally learn a smooth, interpretable feature system for sentence encoding. In addition, since these models do not incorporate a prior over $\vec{z}$, they cannot be used to assign probabilities to sentences or to sample novel sentences. Similarly, Iyyer et al. (2014) provide a method for generating sentences with arbitrary syntactic structure using tree-structured autoencoders, but that model only transforms existing sentences and cannot generate entirely new ones.

Two other models have shown promise in learning sentence encodings, but cannot be used in a generative setting: Skip-thought models (Kiros et al., 2015) are unsupervised learning models that take the same model structure as a sequence autoencoder, but generate text conditioned on a neighboring sentence from the target text, instead of on the target sentence itself. Finally, paragraph vector models (Le and Mikolov, 2014) are non-recurrent sentence representation models. In a paragraph vector model, the encoding of a sentence is obtained by performing gradient-based inference on a prospective encoding vector with the goal of using it to predict the words in the sentence.

2.2 The variational autoencoder

The variational autoencoder (VAE, Kingma and Welling, 2015; Rezende et al., 2014) is a generative model that is based on a regularized version of the standard autoencoder. This model imposes a prior distribution on the hidden codes $\vec{z}$ which enforces a regular geometry over codes and makes it possible to draw proper samples from the model using ancestral sampling.

The VAE modifies the autoencoder architecture by replacing the deterministic function φ_{enc} with a learned posterior *recognition model*, $q(\vec{z}|x)$. This model parametrizes an approximate posterior distribution over $\vec{z}$ (usually a diagonal Gaussian) with a neural network conditioned on x. Intuitively, the VAE learns codes not as single points, but as soft ellipsoidal *regions* in latent space, forcing the codes to fill the space rather than memorizing the training data as isolated codes.

If the VAE were trained with a standard autoencoder's reconstruction objective, it would learn to encode its inputs deterministically by making the variances in $q(\vec{z}|x)$ vanishingly small (Raiko et al., 2015). Instead, the VAE uses an objective which encourages the model to keep its posterior distributions close to a prior $p(\vec{z})$, generally a standard Gaussian ($\mu = \vec{0}$, $\sigma = \vec{1}$). Additionally, this objective is a valid lower bound on the true log likelihood of the data, making the VAE a generative model. This objective takes the following form:

$$\begin{aligned}\mathcal{L}(\theta; x) = &-\mathrm{KL}(q_\theta(\vec{z}|x)||p(\vec{z})) \\ &+ \mathbb{E}_{q_\theta(\vec{z}|x)}[\log p_\theta(x|\vec{z})] \\ \leq &\log p(x) \ .\end{aligned} \tag{1}$$

This forces the model to be able to decode plausible sentences from every point in the latent space that has a reasonable probability under the prior.

In the experiments presented below using VAE models, we use diagonal Gaussians for the prior and posterior distributions $p(\vec{z})$ and $q(\vec{z}|x)$, using the Gaussian reparameterization trick of Kingma and Welling (2015). We train our models with stochastic gradient descent, and at each gradient step we estimate the reconstruction cost using a

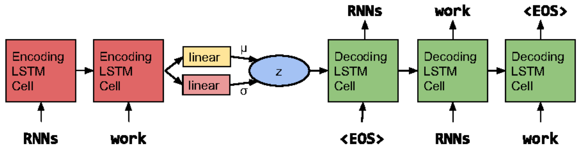

Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned randomly-initialized dictionary of embedding vectors. $\vec{z}$ is a vector-valued latent variable with a Gaussian prior and an approximate posterior parameterized by the encoder's outputs μ and σ. <EOS> marks the end of each sequence.

single sample from $q(\vec{z}|x)$, but compute the KL divergence term of the cost function in closed form, again following Kingma and Welling (2015).

3 A VAE for sentences

We adapt the variational autoencoder to text by using single-layer LSTM RNNs (Hochreiter and Schmidhuber, 1997) for both the encoder and the decoder, essentially forming a sequence autoencoder with the Gaussian prior acting as a regularizer on the hidden code. The decoder serves as a special RNN language model that is conditioned on this hidden code, and in the degenerate setting where the hidden code incorporates no useful information, this model is effectively equivalent to an RNNLM. The model is depicted in Figure 1, and is used in all of the experiments discussed below.

We explored several variations on this architecture, including concatenating the sampled $\vec{z}$ to the decoder input at every time step, using a softplus parametrization for the variance, and using deep feedforward networks between the encoder and latent variable and the decoder and latent variable. We noticed little difference in the model's performance when using any of these variations. However, when including feedforward networks between the encoder and decoder we found that it is necessary to use highway network layers (Srivastava et al., 2015) for the model to learn. We use a 4 layer highway network to parametrize the Gaussian posterior conditioned on the RNN state, and another identical network to map the Gaussian samples back to feed into the decoder RNN. We discuss hyperparameter tuning in Appendix II.

We also experimented with more sophisticated recognition models $q(\vec{z}|x)$, including a multistep sampling model styled after DRAW (Gregor et al., 2015), and a posterior approximation using normalizing flows (Rezende and Mohamed, 2015). However, we were unable to reap significant gains over our plain VAE.

While the strongest results with VAEs to date have been on continuous domains like images, there has been some work on discrete sequences: a technique for doing this using RNN encoders and decoders, which shares the same high-level architecture as our model, was proposed under the name Variational Recurrent Autoencoder (VRAE) for the modeling of music in Fabius and van Amersfoort (2014). While there has been other work on including continuous latent variables in RNN-style models for modeling speech, handwriting, and music (Bayer and Osendorfer, 2015; Chung et al., 2015), these models include separate latent variables per timestep and are unsuitable for our goal of modeling global features.

In a recent paper with goals similar to ours, Miao et al. (2016) introduce an effective VAE-based document-level language model that models texts as bags of words, rather than as sequences. They mention briefly that they have to train the encoder and decoder portions of the network in alternation rather than simultaneously, possibly as a way of addressing some of the issues that we discuss in Section 3.1.

3.1 Optimization challenges

Our model aims to learn global latent representations of sentence content. We can quantify the degree to which our model learns global features by looking at the variational lower bound objective (1). The bound breaks into two terms: the data likelihood under the posterior (expressed as cross entropy), and the KL divergence of the posterior from the prior. A model that encodes useful information in the latent variable $\vec{z}$ will have a non-zero KL divergence term and a relatively small cross entropy term. Straightforward implementations of our VAE fail to learn this behavior: except in vanishingly rare cases, most training runs with most hyperparameters yield models that consistently set $q(\vec{z}|x)$ equal to the prior $p(\vec{z})$, bringing the KL divergence term of the cost function to zero.

When the model does this, it is essentially behaving as an RNNLM. Because of this, it can express arbitrary distributions over the output sentences (albeit with a potentially awkward left-to-right factorization) and can thereby achieve likelihoods that are close to optimal. Previous work on VAEs for image modeling (Kingma and Welling, 2015) used a much weaker independent pixel decoder model $p(x|\vec{z})$, forcing the model to use the global latent variable to achieve good likelihoods. In a related result, recent approaches to image generation that use LSTM decoders are able to do well without VAE-style global latent variables (Theis and Bethge, 2015).

This problematic tendency in learning is compounded by the LSTM decoder's sensitivity to subtle variation in the hidden states, such as that introduced by the posterior sampling process. This

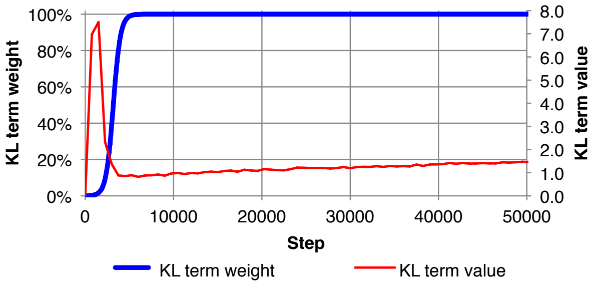

Figure 2: The weight of the KL divergence term of variational lower bound according to a typical sigmoid annealing schedule plotted alongside the (unweighted) value of the KL divergence term for our VAE on the Penn Treebank.

causes the model to initially learn to ignore $\vec{z}$ and go after low hanging fruit, explaining the data with the more easily optimized decoder. Once this has happened, the decoder ignores the encoder and little to no gradient signal passes between the two, yielding an undesirable stable equilibrium with the KL cost term at zero. We propose two techniques to mitigate this issue.

KL cost annealing In this simple approach to this problem, we add a variable weight to the KL term in the cost function at training time. At the start of training, we set that weight to zero, so that the model learns to encode as much information in $\vec{z}$ as it can. Then, as training progresses, we gradually increase this weight, forcing the model to smooth out its encodings and pack them into the region of the embedding space that is assigned a reasonably high probability by the Gaussian prior. We increase this weight until it reaches 1, at which point the weighted cost function is equivalent to the true variational lower bound. In this setting, we do not optimize the proper lower bound on the training data likelihood during the early stages of training, but we nonetheless see improvements on the value of that bound at convergence. This can be thought of as annealing from a vanilla autoencoder to a VAE. The rate of this increase is tuned as a hyperparameter.

Figure 2 shows the behavior of the KL cost term during the first 50k steps of training on Penn Treebank (Marcus et al., 1993) language modeling with KL cost annealing in place. This example reflects a pattern that we observed often: KL spikes early in training while the model can encode information in $\vec{z}$ cheaply, then drops substantially once it begins paying the full KL divergence penalty, and finally slowly rises again before converging as the model learns to condense more information into $\vec{z}$.

Word dropout and historyless decoding In addition to weakening the penalty term on the encodings, we also experiment with weakening the decoder. As in RNNLMs and sequence autoencoders, during learning our decoder predicts each word conditioned on the ground-truth previous word. A natural way to weaken the decoder is to remove some or all of this conditioning information during learning. We do this by randomly replacing some fraction of the conditioned-on word tokens with the generic unknown word token UNK. This forces the model to rely on the latent variable $\vec{z}$ to make good predictions. This technique is a variant of word dropout (Iyyer et al., 2015; Kumar et al., 2016), applied not to a feature extractor but to a decoder. We also experimented with standard dropout (Srivastava et al., 2014) applied to the input word embeddings in the decoder, but this did not help the model learn to use the latent variable.

This technique is parameterized by a keep rate $k \in [0, 1]$. We tune this parameter both for our VAE and for our baseline RNNLM. Taken to the extreme of $k = 0$, the decoder sees no input, and is thus able to condition only on the number of words produced so far, yielding a model that is extremely limited in the kinds of distributions it can model without using $\vec{z}$.

4 Results: Language modeling

In this section, we report on language modeling experiments on the Penn Treebank in an effort to discover whether the inclusion of a global latent variable is helpful for this standard task. For this reason, we restrict our VAE hyperparameter search to those models which encode a non-trivial amount in the latent variable, as measured by the KL divergence term of the variational lower bound.

Results We used the standard train–test split for the corpus, and report test set results in Table 2. The results shown reflect the training and test set performance of each model at the training step at which the model performs best on the development set. Our reported figures for the VAE reflect the variational lower bound on the test likelihood, while for the RNNLMs, which can be evaluated exactly, we report the true test likelihood. This discrepancy puts the VAE at a potential disadvantage.

In the standard setting, the VAE performs slightly worse than the RNNLM baseline, though it does succeed in using the latent space to a limited extent: it has a reconstruction cost (99) better than that of the baseline RNNLM, but makes up for this with a KL divergence cost of 2. Training a VAE in the standard setting without both word dropout and cost annealing reliably results in models with equivalent performance to the baseline RNNLM, and zero KL divergence.

To demonstrate the ability of the latent variable to encode the full content of sentences in addition

Model	Standard				Inputless Decoder			
	Train NLL	Train PPL	Test NLL	Test PPL	Train NLL	Train PPL	Test NLL	Test PPL
RNNLM	100 –	95	**100** –	**116**	135 –	600	135 –	> 600
VAE	98 (2)	100	101 (2)	119	120 (15)	300	**125** (15)	**380**

Table 2: Penn Treebank language modeling results, reported as negative log likelihoods (NLL) and as perplexities (PPL). Lower is better for both metrics. For the VAE, the KL term of the likelihood is shown in parentheses alongside the total likelihood.

to more abstract global features, we also provide numbers for an inputless decoder that does not condition on previous tokens, corresponding to a word dropout keep rate of 0. If this decoder cannot or does not take advantage of the encoder, then it is essentially equivalent to a unigram language model, with the hidden state providing information about position but noting more. In this regime we can see that the variational lower bound contains a significantly larger KL term and shows a substantial improvement over the weakened RNNLM. While it is weaker than a standard decoder, the inputless decoder has the interesting property that its sentence generating process is fully differentiable. Advances in generative models of this kind could be promising as a means of generating text while using adversarial training methods, which require differentiable generators.

Even with the techniques described in the previous section, including the inputless decoder, we were unable to train models for which the KL divergence term of the cost function dominates the reconstruction term. This suggests that it is still substantially easier to learn to factor the data distribution using simple local statistics, as in the RNNLM, such that an encoder will only learn to encode information in $\vec{z}$ when that information cannot be effectively described by these local statistics.

5 Results: Imputing missing words

We claim that the our VAE's global sentence features make it especially well suited to the task of imputing missing words in otherwise known sentences. In this section, we present a technique for imputation and a novel evaluation strategy inspired by adversarial training. Qualitatively, we find that the VAE yields more diverse and plausible imputations for the same amount of computation (see the examples given in Table 3), but precise quantitative evaluation requires intractable likelihood computations. We sidestep this by introducing a novel evaluation strategy.

While the standard RNNLM is a powerful generative model, the sequential nature of likelihood computation and decoding makes it unsuitable for performing inference over unknown words given some known words (the task of *imputation*). Except in the special case where the unknown words all appear at the end of the decoding sequence, sampling from the posterior over the missing variables is intractable for all but the smallest vocabularies. For a vocabulary of size V, it requires $O(V)$ runs of full RNN inference per step of Gibbs sampling or iterated conditional modes. Worse, because of the directional nature of the graphical model given by an RNNLM, many steps of sampling could be required to propagate information between unknown variables and the known downstream variables. The VAE, while it suffers from the same intractability problems when sampling or computing MAP imputations, can more easily propagate information between all variables, by virtue of having a global latent variable and a tractable recognition model.

For this experiment and subsequent analysis, we train our models on the Books Corpus introduced in Kiros et al. (2015). This is a collection of text from 12k e-books, mostly fiction. The dataset, after pruning, contains approximately 80m sentences. We find that this much larger amount of data produces more subjectively interesting generative models than smaller standard language modeling datasets. We use a fixed word dropout rate of 75% when training this model and all subsequent models unless otherwise specified. Our models (the VAE and RNNLM) are trained as language models, decoding right-to-left to shorten the dependencies during learning for the VAE. We use 512 hidden units.

Inference method To generate imputations from the two models, we use beam search with beam size 15 for the RNNLM and approximate iterated conditional modes (Besag, 1986) with 3 steps of a beam size 5 search for the VAE. This allows us to compare the same amount of computation for both models. We find that breaking decoding for the VAE into several sequential steps is necessary to propagate information among the variables. Iterated conditional modes is a technique for finding the maximum joint assignment of a set of variables by alternately maximizing conditional distributions, and is a generalization of "hard-EM" algorithms like k-means (Kearns et al., 1998). For approximate iterated conditional modes, we first initialize the unknown words to the UNK token. We then alternate assigning the latent variable to its mode from the recognition model, and performing

Model	Adv. Err. (%)		NLL
	Unigram	LSTM	RNNLM
RNNLM (15 bm.)	28.3	38.9	46.0
VAE (3x5 bm.)	**22.4**	**35.6**	46.1

Table 4: Results for adversarial evaluation of imputations. Unigram and LSTM numbers are the *adversarial error* (see text) and RNNLM numbers are the negative log-likelihood given to entire generated sentence by the RNNLM, a measure of sentence typicality. Lower is better on both metrics. The VAE is able to generate imputations that are significantly more difficult to distinguish from the true sentences.

constrained beam search to assign the unknown words. Both of our generative models are trained to decode sentences from right-to-left, which shortens the dependencies involved in learning for the VAE, and we impute the final 20% of each sentence. This lets us demonstrate the advantages of the global latent variable in the regime where the RNNLM suffers the most from its inductive bias.

Adversarial evaluation Drawing inspiration from adversarial training methods for generative models as well as non-parametric two-sample tests (Goodfellow et al., 2014; Li et al., 2015b; Denton et al., 2015; Gretton et al., 2012), we evaluate the imputed sentence completions by examining their distinguishability from the true sentence endings. While the non-differentiability of the discrete RNN decoder prevents us from easily applying the adversarial criterion at train time, we can define a very flexible test time evaluation by training a discriminant function to separate the generated and true sentences, which defines an *adversarial error*.

We train two classifiers: a bag-of-unigrams logistic regression classifier and an LSTM logistic regression classifier that reads the input sentence and produces a binary prediction after seeing the final EOS token. We train these classifiers using early stopping on a 80/10/10 train/dev/test split of 320k sentences, constructing a dataset of 50% complete sentences from the corpus (positive examples) and 50% sentences with imputed completions (negative examples). We define the *adversarial error* as the gap between the ideal accuracy of the discriminator (50%, i.e. indistinguishable samples), and the actual accuracy attained.

Results As a consequence of this experimental setup, the RNNLM cannot choose anything outside of the top 15 tokens given by the RNN's initial unconditional distribution $P(x_1|\text{Null})$ when producing the final token of the sentence, since it has not yet generated anything to condition on, and has a beam size of 15. Table 4 shows that this weakness

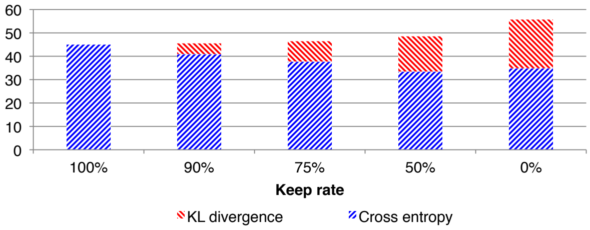

Figure 3: The values of the two terms of the cost function as word dropout increases.

makes the RNNLM produce far less diverse samples than the VAE and suffer accordingly versus the adversarial classifier. Additionally, we include the score given to the entire sentence with the imputed completion given a separate independently trained language model. The likelihood results are comparable, though the RNNLMs favoring of generic high-probability endings such as "he said," gives it a slightly lower negative log-likelihood. Measuring the RNNLM likelihood of sentences themselves produced by an RNNLM is not a good measure of the power of the model, but demonstrates that the RNNLM can produce what it sees as high-quality imputations by favoring typical local statistics, even though their repetitive nature produces easy failure modes for the adversarial classifier. Accordingly, under the adversarial evaluation our model substantially outperforms the baseline since it is able to efficiently propagate information bidirectionally through the latent variable.

6 Analyzing variational models

We now turn to more qualitative analysis of the model. Since our decoder model $p(x|\vec{z})$ is a sophisticated RNNLM, simply sampling from the directed graphical model (first $p(\vec{z})$ then $p(x|\vec{z})$) would not tell us much about how much of the data is being explained by each of the latent space and the decoder. Instead, for this part of the evaluation, we sample from the Gaussian prior, but use a greedy deterministic decoder for $p(x|\vec{z})$, the RNNLM conditioned on $\vec{z}$. This allows us to get a sense of how much of the variance in the data distribution is being captured by the distributed vector $\vec{z}$ as opposed to the decoder. Interestingly, these results qualitatively demonstrate that large amounts of variation in generated language can be achieved by following this procedure. In Appendix I, we provide some results on small text classification tasks.

6.1 Analyzing the impact of word dropout

For this experiment, we train on the Books Corpus and test on a held out 10k sentence test set from that corpus. We find that train and test set performance are very similar. In Figure 3, we examine the impact of word dropout on the variational lower bound, broken down into KL divergence and cross entropy components. We drop out words with the specified keep rate at training time, but supply all words as inputs at test time except in the 0% setting.

We do not re-tune the hyperparameters for each run, which results in the model with no dropout encoding very little information in $\vec{z}$ (i.e., the KL component is small). We can see that as we lower the keep rate for word dropout, the amount of information stored in the latent variable increases, and the overall likelihood of the model degrades somewhat. Results from the Section 4 indicate that a model with no latent variable would degrade in performance significantly more in the presence of heavy word dropout.

We also qualitatively evaluate samples, to demonstrate that the increased KL allows meaningful sentences to be generated purely from continuous sampling. Since our decoder model $p(x|\vec{z})$ is a sophisticated RNNLM, simply sampling from the directed graphical model (first $p(\vec{z})$ then $p(x|\vec{z})$) would not tell us about how much of the data is being explained by the learned vector vs. the language model. Instead, for this part of the qualitative evaluation, we sample from the Gaussian prior, but use a greedy deterministic decoder for x, taking each token $x_t = \text{argmax}_{x_t} p(x_t|x_{0,\ldots,t-1}, \vec{z})$. This allows us to get a sense of how much of the variance in the data distribution is being captured by the distributed vector $\vec{z}$ as opposed to by local language model dependencies.

These results, shown in Table 5, qualitatively demonstrate that large amounts of variation in generated language can be achieved by following this procedure. At the low end, where very little of the variance is explained by $\vec{z}$, we see that greedy decoding applied to a Gaussian sample does not produce diverse sentences. As we increase the amount of word dropout and force $\vec{z}$ to encode more information, we see the sentences become more varied, but past a certain point they begin to repeat words or show other signs of ungrammaticality. Even in the case of a fully dropped-out decoder, the model is able to capture higher-order statistics not present in the unigram distribution.

Additionally, in Table 6 we examine the effect of using lower-probability samples from the latent Gaussian space for a model with a 75% word keep rate. We find lower-probability samples by applying an approximately volume-preserving transformation to the Gaussian samples that stretches some eigenspaces by up to a factor of 4. This has the effect of creating samples that are not too improbable under the prior, but still reach into the tails of the distribution. We use a random linear transformation, with matrix elements drawn from a uniform distribution from $[-c, c]$, with c chosen to give the desired properties (0.1 in our experiments). Here we see that the sentences are far less typical, but for the most part are grammatical and maintain a clear topic, indicating that the latent variable is capturing a rich variety of global features even for rare sentences.

6.2 Sampling from the posterior

In addition to generating unconditional samples, we can also examine the sentences decoded from the posterior vectors $p(\vec{z}|x)$ for various sentences x. Because the model is regularized to produce distributions rather than deterministic codes, it does not exactly memorize and round-trip the input. Instead, we can see what the model considers to be similar sentences by examining the posterior samples in Table 7. The codes appear to capture information about the number of tokens and parts of speech for each token, as well as topic information. As the sentences get longer, the fidelity of the round-tripped sentences decreases.

6.3 Homotopies

The use of a variational autoencoder allows us to generate sentences using greedy decoding on continuous samples from the space of codes. Additionally, the volume-filling and smooth nature of the code space allows us to examine for the first time a concept of *homotopy* (linear interpolation) between sentences. In this context, a homotopy between two codes $\vec{z}_1$ and $\vec{z}_2$ is the set of points on the line between them, inclusive, $\vec{z}(t) = \vec{z}_1 * (1-t) + \vec{z}_2 * t$ for $t \in [0, 1]$. Similarly, the homotopy between two sentences decoded (greedily) from codes $\vec{z}_1$ and $\vec{z}_2$ is the set of sentences decoded from the codes on the line. Examining these homotopies allows us to get a sense of what neighborhoods in code space look like – how the autoencoder organizes information and what it regards as a continuous deformation between two sentences.

While a standard non-variational RNNLM does not have a way to perform these homotopies, a vanilla sequence autoencoder can do so. As mentioned earlier in the paper, if we examine the homotopies created by the sequence autoencoder in Table 1, though, we can see that the transition between sentences is sharp, and results in ungrammatical intermediate sentences. This gives evidence for our intuition that the VAE learns representations that are smooth and "fill up" the space.

In Table 8 (and in Table 12 in Appendix III, which shows additional homotopies) we can see that the codes mostly contain syntactic information, such as the number of words and the parts of speech of tokens, and that all intermediate sentences are grammatical. Some topic information also remains consistent in neighborhoods along the path. Additionally, sentences with similar syntax and topic but flipped sentiment valence, e.g.

“ i want to talk to you . ”
“i want to be with you . ”
“i do n’t want to be with you . ”
i do n’t want to be with you .
she did n’t want to be with him .
he was silent for a long moment .
he was silent for a moment .
it was quiet for a moment .
it was dark and cold .
there was a pause .
it was my turn .
there is no one else in the world .
there is no one else in sight .
they were the only ones who mattered .
they were the only ones left .
he had to be with me .
she had to be with him .
i had to do this .
i wanted to kill him .
i started to cry .
i turned to him .
no .
he said .
“ no , ” he said .
“ no , ” i said .
“ i know , ” she said .
“ thank you , ” she said .
“ come with me , ” she said .
“ talk to me , ” she said .
“ do n’t worry about it , ” she said .

Table 8: Paths between pairs of random points in VAE space: Note that intermediate sentences are grammatical, and that topic and syntactic structure are usually locally consistent.

“the pain was unbearable” vs. “the thought made me smile”, can have similar embeddings, a phenomenon which has been observed with single-word embeddings (for example the vectors for “bad” and “good” are often very similar due to their similar distributional characteristics).

7 Conclusion

This paper introduces the use of a variational autoencoder for natural language sentences. We present novel techniques that allow us to train our model successfully, and find that it can effectively impute missing words. We analyze the latent space learned by our model, and find that it is able to generate coherent and diverse sentences through purely continuous sampling and provides interpretable homotopies that smoothly interpolate between sentences.

We hope in future work to investigate factorization of the latent variable into separate style and content components, to generate sentences conditioned on extrinsic features, to learn sentence embeddings in a semi-supervised fashion for language understanding tasks like textual entailment, and to go beyond adversarial evaluation to a fully adversarial training objective.

Method	Accuracy	F1
Feats	73.2	–
RAE+DP	72.6	–
RAE+feats	74.2	–
RAE+DP+feats	76.8	83.6
ST	73.0	81.9
Bi-ST	71.2	81.2
Combine-ST	73.0	82.0
VAE	72.9	81.4
VAE+feats	75.0	82.4
VAE+combine-ST	74.8	82.3
Feats+combine-ST	75.8	83.0
VAE+combine-ST+feats	**76.9**	**83.8**

Table 9: Results for the MSR Paraphrase Corpus.

Appendix I: Text classification

In order to further examine the the structure of the representations discovered by the VAE, we conduct classification experiments on paraphrase detection and question type classification. We train a VAE with a hidden state size of 1200 hidden units on the Books Corpus, and use the posterior mean of the model as the extracted sentence vector. We train classifiers on these means using the same experimental protocol as Kiros et al. (2015).

Paraphrase detection For the task of paraphrase detection, we use the Microsoft Research Paraphrase Corpus (Dolan et al., 2004). We compute features from the sentence vectors of sentence pairs in the same way as Kiros et al. (2015), concatenating the elementwise products and the absolute value of the elementwise differences of the two vectors. We train an ℓ_2-regularized logistic regression classifier and tune the regularization strength using cross-validation.

We present results in Table 9 and compare to several previous models for this task. *Feats* is the lexicalized baseline from Socher et al. (2011). RAE uses the recursive autoencoder from that work, and DP adds their dynamic pooling step to calculate pairwise features. ST uses features from the unidirectional skip-thought model, *bi*-ST uses bidirectional skip-thought, and *combine*-ST uses the concatenation of those features. We also experimented with concatenating lexical features and the two types of distributed features.

We found that our features performed slightly worse than skip-thought features by themselves and slightly better than recursive autoencoder features, and were complementary and yielded strong performance when simply concatenated with the skip-thought features.

Method	Accuracy
ST	91.4
Bi-ST	89.4
Combine-ST	**92.2**
AE	84.2
VAE	87.0
CBOW	87.3
VAE, combine-ST	92.0
RNN	90.2
CNN	**93.6**

Table 10: Results for TREC Question Classification.

	Standard		Inputless Decoder	
	RNNLM	VAE	RNNLM	VAE
D_{word}	464	353	305	499
D_{LSTM}	337	191	68	350
D_z	–	13	–	111
KR	0.66	0.62	–	–

Table 11: Automatically selected hyperparameter values used for the models used in the Penn Treebank language modeling experiments. KR is the keep rate for word dropout.

Question classification We also conduct experiments on the TREC Question Classification dataset of Li and Roth (2002). Following Kiros et al. (2015), we train an ℓ_2-regularized softmax classifier with 10-fold cross-validation to set the regularization. Note that using a linear classifier like this one may disadvantage our representations here, since the Gaussian distribution over hidden codes in a VAE is likely to discourage linear separability.

We present results in Table 10. Here, AE is a plain sequence autoencoder. We compare with results from a bag of word vectors (CBOW, Zhao et al., 2015) and skip-thought (ST). We also compare with an RNN classifier (Zhao et al., 2015) and a CNN classifier (Kim, 2014) both of which, unlike our model, are optimized end-to-end. We were not able to make the VAE codes perform better than CBOW in this case, but they did outperform features from the sequence autoencoder. Skip-thought performed quite well, possibly because the skip-thought training objective of next sentence prediction is well aligned to this task: it essentially trains the model to generate sentences that address implicit open questions from the narrative of the book. Combining the two representations did not give any additional performance gain over the base skip-thought model.

Appendix II: Hyperparameter tuning

We extensively tune the hyperparameters of each model using an automatic Bayesian hyperparameter tuning algorithm (based on Snoek et al., 2012) over development set data. We run the model with each set of hyperpameters for 10 hours, operating 12 experiments in parallel, and choose the best set of hyperparameters after 200 runs. Results for our language modeling experiments are reported in Table 11 on the next page.

Appendix III: Additional homotopies

Table 12 shows additional homotopies from our model. We observe that intermediate sentences are almost always grammatical, and often contain consistent topic, vocabulary and syntactic information in local neighborhoods as they interpolate between the endpoint sentences. Because the model is trained on fiction, including romance novels, the topics are often rather dramatic.

Acknowledgments

We thank the Google Brain team, Alireza Makhzani, Laurent Dinh, Jon Gauthier, the Stanford NLP Group, and our anonymous reviewers for their helpful contributions and feedback.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *Proc. ICLR*.

Justin Bayer and Christian Osendorfer. 2015. Learning stochastic recurrent networks. arXiv preprint arXiv:1411.7610.

Julian Besag. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society Series B (Methodological)* pages 48–259.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Proc. NIPS*.

Andrew M. Dai and Quoc V. Le. 2015. Semi-supervised sequence learning. In *Proc. NIPS*.

Emily Denton, Soumith Chintala, Arthur Szlam, and Rob Fergus. 2015. Deep generative image models using a laplacian pyramid of adversarial networks. In *Proc. NIPS*.

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

Jeff Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini

amazing , is n’t it ?
so , what is it ?
it hurts , isnt it ?
why would you do that ?
“ you can do it .
“ i can do it .
i ca n’t do it .
“ i can do it .
“ do n’t do it .
“ i can do it .
i could n’t do it .
i dont like it , he said .
i waited for what had happened .
it was almost thirty years ago .
it was over thirty years ago .
that was six years ago .
he had died two years ago .
ten , thirty years ago .
“ it ’s all right here .
“ everything is all right here .
“ it ’s all right here .
it ’s all right here .
we are all right here .
come here in five minutes .
this was the only way .
it was the only way .
it was her turn to blink .
it was hard to tell .
it was time to move on .
he had to do it again .
they all looked at each other .
they all turned to look back .
they both turned to face him .
they both turned and walked away .
im fine .
youre right .
“ all right .
you ’re right .
okay , fine .
“ okay , fine .
yes , right here .
no , not right now .
“ no , not right now .
“ talk to me right now .
please talk to me right now .
i ’ll talk to you right now .
“ i ’ll talk to you right now .
“ you need to talk to me now .
“ but you need to talk to me now .

Table 12: Selected homotopies between pairs of random points in the latent VAE space.

Venugopalan, Kate Saenko, and Trevor Darrell. 2015. Long-term recurrent convolutional networks for visual recognition and description. In *Proc. CVPR*.

Otto Fabius and Joost R. van Amersfoort. 2014. Variational recurrent auto-encoders. arXiv preprint arXiv:1412.6581.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Proc. NIPS*.

Karol Gregor, Ivo Danihelka, Alex Graves, and Daan Wierstra. 2015. DRAW: A recurrent neural network for image generation. In *Proc. ICML*.

Arthur Gretton, Karsten M Borgwardt, Malte J Rasch, Bernhard Schölkopf, and Alexander Smola. 2012. A kernel two-sample test. *JMLR* 13(1):723–773.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9(8).

Mohit Iyyer, Jordan Boyd-Graber, and Hal Daumé III. 2014. Generating sentences from semantic vector space representations. In *Proc. NIPS Workshop on Learning Semantics*.

Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. 2015. Deep unordered composition rivals syntactic methods for text classification. In *Proc. ACL*.

Michael Kearns, Yishay Mansour, and Andrew Y Ng. 1998. An information-theoretic analysis of hard and soft assignment methods for clustering. In *Learning in graphical models*, Springer, pages 495–520.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proc. EMNLP*.

Diederik P. Kingma and Max Welling. 2015. Auto-encoding variational bayes. In *Proc. ICLR*.

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

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2016. Ask me anything: Dynamic memory networks for natural language processing. In *Proc. ICML*.

Quoc V. Le and Tomáš Mikolov. 2014. Distributed representations of sentences and documents. In *Proc. ICML*.

Jiwei Li, Minh-Thang Luong, and Dan Jurafsky. 2015a. A hierarchical neural autoencoder for paragraphs and documents. In *Proc. ACL*.

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

Yujia Li, Kevin Swersky, and Richard Zemel. 2015b. Generative moment matching networks. In *Proc. ICML*.

Junhua Mao, Wei Xu, Yi Yang, Jiang Wang, Zhiheng Huang, and Alan Yuille. 2015. Deep captioning with multimodal recurrent neural networks (m-RNN). In *Proc. ICLR*.

Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics* 19(2):313–330.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *Proc. ICML*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proc. ICASSP*.

Tapani Raiko, Mathias Berglund, Guillaume Alain, and Laurent Dinh. 2015. Techniques for learning binary stochastic feedforward neural networks. In *Proc. ICLR*.

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

Danilo J. Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proc. ICML*.

Jasper Snoek, Hugo Larochelle, and Ryan P. Adams. 2012. Practical Bayesian optimization of machine learning algorithms. In *Proc. NIPS*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D. Manning, and Andrew Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proc. NIPS*.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: A simple way to prevent neural networks from overfitting. *JMLR* 15(1):1929–1958.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. 2015. Training very deep networks. In *Proc. NIPS*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proc. NIPS*.

Lucas Theis and Matthias Bethge. 2015. Generative image modeling using spatial LSTMs. In *Proc. NIPS*.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. 2015. Show and tell: A neural image caption generator. In *Proc. CVPR*.

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

but now , as they parked out front and owen stepped out of the car , he could see _ _ _ _ _ _		
True: *that the transition was complete .*	**RNNLM:** *it , " i said .*	**VAE:** *through the driver 's door .*
you kill him and his _ _		
True: *men .*	**RNNLM:** *. "*	**VAE:** *brother .*
not surprising , the mothers dont exactly see eye to eye with me _ _ _ _		
True: *on this matter .*	**RNNLM:** *, i said .*	**VAE:** *, right now .*
outside the cover , quiet _ _		
True: *fell .*	**RNNLM:** *. "*	**VAE:** *time .*
she punched the cell _ _		
True: *too .*	**RNNLM:** *again .*	**VAE:** *phone .*

Table 3: Examples of using beam search to impute missing words within sentences. Since we decode from right to left, note the stereotypical completions given by the RNNLM, compared to the VAE completions that often use topic data and more varied vocabulary.

100% word keep	**75% word keep**
" no , " he said .	*why would i want you to look at me like this ?*
" no , " he said .	*" love you , too . "*
" thank you , " he said .	*she put her hand on his shoulder and followed him to the door .*
50% word keep	**0% word keep**
all this time , i could n't stay in the room .	*not , did n't be , for the time he was out in*
" maybe two or two . "	*i i hear some of of of*
she laughed again , once again , once again , and thought about it for a moment in long silence .	*i was noticed that she was holding the in in of the the in*

Table 5: Samples from a model trained with varying amounts of word dropout. We sample a vector from the Gaussian prior and apply greedy decoding to the result. Note that diverse samples can be achieved using a purely deterministic decoding procedure. Once we use reach a purely inputless decoder in the 0% setting, however, the samples cease to be plausible English sentences.

he had been unable to conceal the fact that there was a logical explanation for his inability to alter the fact that they were supposed to be on the other side of the house .
with a variety of pots strewn scattered across the vast expanse of the high ceiling , a vase of colorful flowers adorned the tops of the rose petals littered the floor and littered the floor .
atop the circular dais perched atop the gleaming marble columns began to emerge from atop the stone dais, perched atop the dais .

Table 6: Greedily decoded sentences from a model with 75% word keep probability, sampling from lower-likelihood areas of the latent space. Note the consistent topics and vocabulary usage.

INPUT	**we looked out at the setting sun .**	**i went to the kitchen .**	**how are you doing ?**
MEAN	*they were laughing at the same time .*	*i went to the kitchen .*	*what are you doing ?*
SAMP. 1	*ill see you in the early morning .*	*i went to my apartment .*	*" are you sure ?*
SAMP. 2	*i looked up at the blue sky .*	*i looked around the room .*	*what are you doing ?*
SAMP. 3	*it was down on the dance floor .*	*i turned back to the table .*	*what are you doing ?*

Table 7: Three sentences which were used as inputs to the VAE, presented with greedy decodes from the mean of the posterior distribution, and from three samples from that distribution.