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.

Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning (CoNLL), pages 10–21,
Berlin, Germany, August 7-12, 2016.

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:

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

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

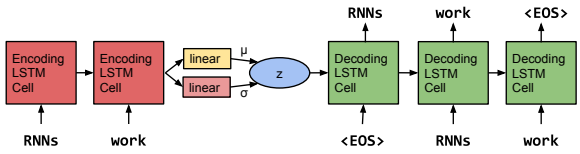

Figure 1: The core structure of our variational autoencoder language model. Words are represented using a learned randomly-initialized dictionary of embedding vectors. $\vec{z}$ is a vector-valued latent variable with a Gaussian prior and an approximate posterior parameterized by the encoder's outputs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

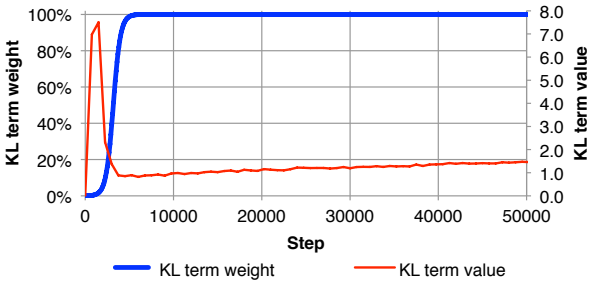

Figure 2: The weight of the KL divergence term of variational lower bound according to a typical sigmoid annealing schedule plotted alongside the (unweighted) value of the KL divergence term for our VAE on the Penn Treebank.

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