Deterministic and Deep Generative Language Models

Kai Liang
University of Amsterdam
1012 WX, Amsterdam

Billy Chan
University of Amsterdam
1012 WX, Amsterdam

Yijie Zhang
University of Amsterdam
1012 WX, Amsterdam

{kai.liang, billy.chan, yijie.zhang}@student.uva.nl

Abstract

In this study, a deterministic recurrent neural network language model (RNN-LM) is implemented and compared with a generative variational autoencoder (VAE) model. These two models are trained and tested on the Penn Treebank (PTB) dataset and evaluated with different metrics (like perplexity). The final results indicate that the VAE model achieves better performance except for having higher complexity, and it allows for approximating the data distribution.

1 Introduction

Existing methods of implementing language models consist of both deterministic and generative approaches. Deterministic models such as RNN-LM (Mikolov et al., 2010) are generally easier to train, while they generate sentences one word at a time. On the other hand, emerging techniques like VAE-based language models (Bowman et al., 2015) are generative models which learn the probability distribution of data and model sentences as a whole in meaningful latent dimensions. In this project, the two types of language models are implemented and evaluated with respect to different metrics, including negative log-likelihood, sentence perplexity, word prediction accuracy, and negative ELBO (for VAE). Furthermore, different sampling strategies are applied to examine the generation quality of the two models.

2 Model[1]

2.1 RNN-LM

The architecture we use for the RNN-LM is shown in Figure 1. In short, sentences passed through the model are processed sequentially. Beginning with the initial hidden state $\mathbf{h}_0$, the LSTM cell (Cherian et al., 2018) processes one word of every sentence in a mini-batch at each time step t, and updates the hidden states $\mathbf{h}_t$ accordingly. As the network is recurrent, it can generate an output at each time step. Hence, the goal of the model is to correctly predict the next word in each sentence at every time step.

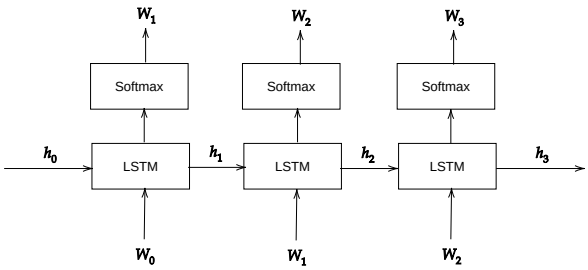

Figure 1: Visualization of the main architecture of RNN-LM.

2.2 Sentence VAE

2.2.1 Theoretical Background

For a generative model with data x and latent variable z, the objective is to maximize the probability of the data given the model:

$$p(x) = \int p(x|z)p(z)dz$$

However, as $p(x)$ is generally intractable, we need to compute it approximately by sampling z, while another problem is, for most sampled z, $p(x|z)$ is close to 0, which does not contribute to the computation. Hence, we want to sample from $p(z|x)$, which is also intractable but can be approximated by a learnable function $q(z|x)$. Then, for a model parameterized by θ and ϕ, we can rewrite the posterior in terms of $q_\phi(z|x)$. By Jensen's Inequality, the logarithm of an expectation is lower bounded by the expectation of the logarithm, leading to the

[1] GitHub repository: https://github.com/s2948044/Natural-Language-Processing-2/tree/master/week2

Evidence Lower Bound (ELBO) that we can optimize on. Note that ELBO is equivalently:

$$-\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}\right] + \log p_\theta(x)$$

in which the first term is $\mathrm{KL}(q_\phi(z|x)\|p_\theta(z|x))$. Hence, the data log-likelihood is:

$$\begin{aligned}\log p_\theta(x) &= \log \int q_\phi(z|x)\frac{p_\theta(x,z)}{q_\phi(z|x)}\mathrm{d}z \\ &= \log\left(\mathbb{E}_{q_\phi(z|x)}\left[\frac{p_\theta(x,z)}{q_\phi(z|x)}\right]\right) \\ &\geq \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right]\end{aligned}$$

and with further re-arrangements, we end up with the equation below:

$$\begin{aligned}\log p_\theta(x) \geq\ & \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x|z)\right] \\ & - \mathrm{KL}\left(q_\phi(z|x)\|p_\theta(z)\right)\end{aligned}$$

Another technical detail for the implementation is to use the reparameterization trick, such that the network ϕ remains differentiable during backpropagation through μ and Σ, and that the randomness of z is otherwise caused by a auxiliary noise variable ϵ, i.e. $z = g_\phi(\epsilon, x)$.

2.2.2 VAE-LM

The visualization of architecture for the VAE-LM is shown in Figure 2. The model consists of an encoder and a decoder, which are respectively a bidirectional LSTM and unidirectional LSTM. Specifically, each sentence $\mathbf{x}_n$ within the N sentences in the data is firstly encoded by the trainable embedding layer, and passed into the the bi-LSTM. Then, the output of the final hidden state of the encoder is transformed into the distribution parameters of $q(\boldsymbol{z}|\mathbf{x}_n)$ (i.e. $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$). Then, a latent variable $\boldsymbol{z}$ is sampled from $q(\boldsymbol{z}|\mathbf{x}_n)$ with the reparameterization trick. The decoder then takes in the multiple sampled $\boldsymbol{z}$ as the hidden states initialization of the first LSTM cell. New outputs from hidden states are generated from each of this decoder cells. $\mathbf{x}_n$ could then be reconstructed by mapping these outputs with softmax and sampling.

3 Experiments

3.1 Data

The Penn Treebank (PTB) dataset[2] is used for the experiments in this study, which consists of annotated tree-structured sentences. The dataset is separated into three splits namely for training, validation and testing, and the concrete division is 39832 sentences, 1700 sentences and 2416 sentences.

[2]https://catalog.ldc.upenn.edu/LDC99T42

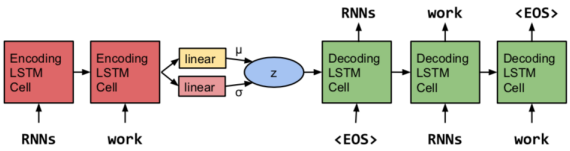

Figure 2: Visualization of the main architecture of VAE-LM (Bowman et al., 2015).

3.2 Preprocessing

Before experiments, the dataset is preprocessed to exclude the syntactic trees of sentences. Then, the 'SOS' and 'EOS' tokens are inserted to each sentence to mark the beginning and end of sentences. Besides, the 'UNK' token is included in the vocabulary to account for words that are not in the training data. Moreover, to allow parallel processing of sentences during training and to accelerate the optimization process, sentences in the mini-batch are padded with the 'PAD' tokens when needed. Thus, the final vocabulary size of our dataset is 44393.

3.3 Experimental Setup

3.3.1 RNN-LM

The model is trained on the training set for 10K iterations, during which the negative log-likelihood, sentence perplexity as well as the prediction accuracy on both the training set and validation set are recorded each 100 steps. After training, the model with the lowest validation perplexity is selected as the final model for testing on the test data and sentence sampling. The main hyperparameter settings of the RNN-LM are shown in Table 1.

Table 1: Hyperparameter settings of RNN-LM.

RNN Type	LSTM	Layer Number	2
Batch Size	25	Hidden Size	128
Learning Rate	0.002	Optimizer	Adam

3.3.2 Sentence VAE

The Sentence VAE model is trained on the training set for 13K steps, during which the NLL, sentence perplexity, prediction accuracy and negative ELBO are measured on the training data and validation set for each 1000 steps. After training, the

model with the lowest validation perplexity is selected as the final model for testing on the test data and sentence sampling. The main hyperparameter settings of the model are shown in Table 2.

Table 2: Hyperparameter settings of VAE-LM.

RNN Type	LSTM	Layer Number	1
Direction Number	1 or 2	Latent Size	64
Batch Size	20	Hidden Size	128
Learning Rate	0.002	Optimizer	Adam

4 Results

4.1 RNN-LM

The performance of the RNN-LM during training with respect to NLL, sentence perplexity and prediction accuracy on the training set and validation set is shown in Figure 3.

As can be inferred from Figure 3, the validation perplexity of the RNN-LM decreases significantly at the early phase of training, which then becomes steady at around 200. However, it climbs up again after 8000 steps. On the other hand, the word prediction accuracy on the validation data undergoes a smoother increase during training, and it eventually stabilizes at around 0.26.

In testing time, the best RNN-LM during training is evaluated on the test data, and the results are shown in Table 3. Moreover, the model is assessed in terms of the generation quality with both greedy decoding (Sentence 1) and random decoding (Sentence 2 to 11) as listed below.

1. The company said the company's earnings rose to $4 million from $1.02 billion.
2. Separately, Manville said it completed the final meeting as the U.S. were troubled over next year, rising spending from the federal claims of CBS, which had studied Bank One authorization.
3. Why these estimates have already gone for inches.
4. The Treasury issue is like passed by now between the talks at overall business.
5. That's this is the Senate. ”
6. They're being concerned with taxes in the statement to keep some wrongdoing; courts burned prices.
7. Laserscope also calls a special quarterly dividend in the second quarter, as much as $5 to $ save a share, or previous a profit of a year earlier, with other net income of $285 million, from $122.7 million.
8. Some important concerns that our cards is buying way.
9. These features spewing raises in time that last morning to improve access land should.
10. I must step to steal the Fed.

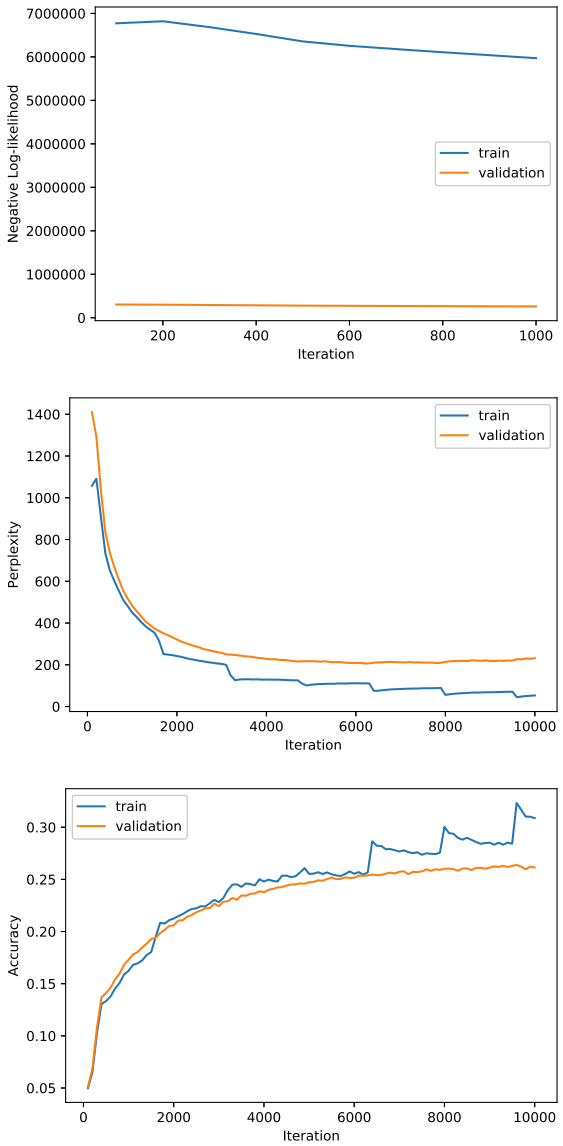

Figure 3: The evolution of NLL, sentence perplexity and prediction accuracy on the training and validation set in terms of training steps for RNN-LM.

11. The new scene became likely to buy China to overseas stocks to buy 8%.

The above sentences generated by the RNN-LM are problematic occasionally in terms of grammar. For instance, redundant punctuation '`''` is generated in Sentence 5, and the '`is`' token is generated after a plural noun '`cards`' in Sentence 8. In addition, due to the essence that the model can only generate one word at each time step for a sentence, these sentences lack coherent meanings in a global sense.

Table 3: The performance of RNN-LM and Sentence VAE model with respect to recorded metrics on the test data.

	NLL	Perplexity	Accuracy	N-ELBO
RNN-LM	308548.9	209.97	0.244	N/A
VAE-LM	318380.3	222.26	0.264	141.81

4.2 Sentence VAE

The performance of the VAE-LM during training with respect to NLL, sentence perplexity, prediction accuracy and negative ELBO on the training set and validation set is shown in Figure 4.

As can be inferred from Figure 4, the validation perplexity of the VAE-LM decreases significantly at the early phase of training, which then becomes steady at around 200. However, it climbs up again after 4000 steps. On the other hand, the word prediction accuracy on the validation data undergoes a smoother increase during training, and it finally stabilizes at around 0.26. It is observable from the graphs that the model has experienced overfitting, in that the training metrics keep continuously improving, while the ones for the validation set drop again after some time.

In testing phase, the best model during training is evaluated with the test data, and the results are displayed in Table 3. Besides, similar to the RNN-LM, the model is assessed in terms of the generation quality by greedy decoding (i.e. Sentence 1 to 7). Linear interpolation is also performed to show the homotopy between two sentences. For instance, below $z_{11} = 0.5 * z_9 + 0.5 * z_{10}$. Besides, the reconstruction capability of the model is tested by reconstructing the Sentence 8 from the test dataset using the approximate posterior mean and 2 samples (Sentence 12 and 13) due to space limitations[3].

[3] others available in GitHub

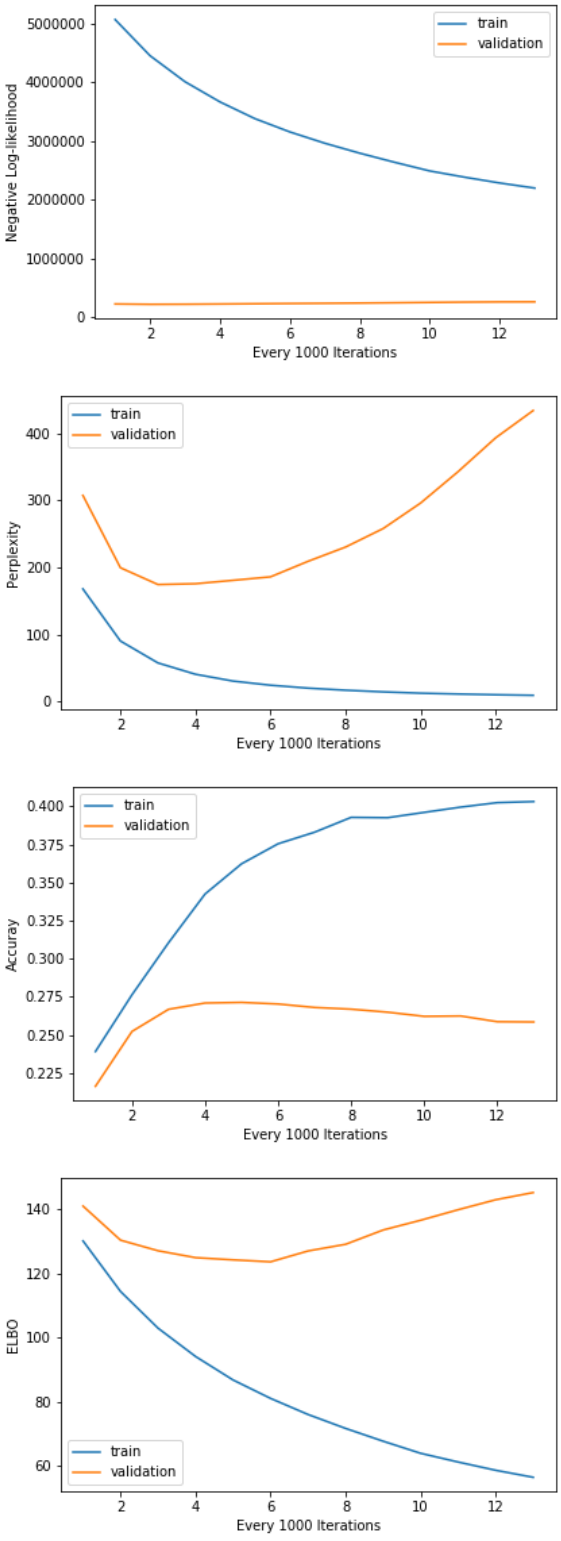

Figure 4: The evolution of the NLL, sentence perplexity, prediction accuracy, and negative ELBO on the training and validation set in terms of training steps for VAE-LM.

1. The fund started selling in the U.S. , possibly as a result of the market .
2. The company 's earnings drop in the third quarter , reflecting higher costs and tightened demand for the third quarter .
3. The one-inch musician mount merit $ 67 million owed to buy UAL shares , which had been expected.
4. The company 's earnings were hurt by a $ 69.8 million gain from the sale of its own Valley unit .
5. the largest detective of the world' s largest private-sector giants.
6. The husbands take effect on the New York Stock Exchange , which has been a symbol of the nation 's largest institutional holders said .
7. The U.S. dollars were introduced by the end of the year .
8. The 13-nation group's net income for the quarter ended Sept. 30, partly because of the company's sales.
9. The board member said, "We don't have a real value of the company."
10. L.J. Hooker, based in Atlanta, is operating with protection from its creditors under Chapter 11 of the U.S. Bankruptcy Code.
11. The one-inch musician share expires at the end of the week.
12. The awake bran nicknames paint the other hand, he says.
13. The group's first acknowledgement of the country's largest opposition, said it will take a beating.

The above sentences generated by the VAE-LM are much better in their quality than the ones generated by the RNN-LM. As could be inferred from these sentences, they are not only more natural and coherent in terms of language usage, but also making fewer grammatical mistakes.

5 Bonus

5.1 The Posterior Collapse Problem

From a high-level perspective, the problem of posterior collapse could be described by the following equation:

$$q_\phi(\boldsymbol{z}|\boldsymbol{x}) \simeq p_\theta(\boldsymbol{z})$$

where the posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ collapses to the prior distribution $q_\phi(\boldsymbol{z})$. In other words, the encoder is not able to extract useful information from the data and learn the parameters of posterior distribution. In this sense, it becomes a problem as the decoder may fail to exploit the latent variable $\boldsymbol{z}$ and hence produce reconstructed data $\tilde{\mathbf{x}}$ that is independent of $\boldsymbol{z}$. It can also be expressed mathematically that the KL term $D_{KL}\left(q_\phi(\boldsymbol{z}|\boldsymbol{x}\|p_\theta(\boldsymbol{z})\right)$ vanishes, so this problem is also called "KL vanishing".

5.2 Methods to Tackle Posterior Collapse

There are different methods to handle the posterior collapse problem, which include, for example, KL annealing, word dropout and KL free-bits.

For the KL annealing (Bowman et al., 2015), its mechanism is placing a weight factor into the KL-term in the objective function, which is initialized to 0 when training starts and increased slowly to 1 as the training proceeds. In this sense, it prevents the posterior distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ from early convergence to $p_\theta(\boldsymbol{z})$, and hence, learning from data is encouraged.

KL free-bits shares the same spirit with KL annealing, but with a different approach. In order to encourage the encoder to learn information from data, threshold ϵ is set for every dimension in the latent vector $\boldsymbol{z}$ in the KL-term, and the KL-term of a dimension will be optimized only when value of it exceeds the threshold. It can be expressed in the modified KL-term as following equation:

$$\sum_{i=1}^{D} \max\left(KL\left(q_\phi\left(z_i|\boldsymbol{x}\right)\|p_\theta\left(z_i\right)\right), \varepsilon\right)$$

The method of word dropout, on the other hand, is aiming at hardening the decoder (Bowman et al., 2015). Hence, during decoding, some of the words in a generated sequence are randomly set to 'UNK' for the next words to condition on. In this way, the model has to rely on $\boldsymbol{z}$ to make better predictions.

6 Conclusion

In previous sections two types of language models are implemented and evaluated. We begin with the deterministic RNN-LM and the results prove that though it is a quite powerful model with relatively lower model complexity, it lacks the capability of generating grammatical and coherent sentences as its word sampling is sequential. On the other hand, the generative VAE model is able to learn not only the probability distribution of data, but also a more meaningful latent space. Consequently, the VAE-LM could generate more diverse and coherent sentences.

However, there are also some limitations in our project. Firstly, due to the limitation in our computational resources, the hyperparameters of the two models are set on an empirical basis, which might lead to imperfect results. In addition, the sentence generations in our implementation is performed in the testing phase, thus, it is unclear how the generated quality evolves during training.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating Sentences from a Continuous Space. *arXiv e-prints* page arXiv:1511.06349.

Thomas Cherian, Akshay Badola, and Vineet Padmanabhan. 2018. Multi-cell LSTM Based Neural Language Model. *arXiv e-prints* page arXiv:1811.06477.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Honza Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH*.