Mitigating Posterior Collapse in Variational Autoencoders for Text

Lennert Jansen
University of Amsterdam
lennertjansen95@gmail.com

Daniël Nobbe
University of Amsterdam
daniellnobbe@gmail.com

Daniel Laszlo
University of Amsterdam
daniel.laszlo94@gmail.com

1 Introduction

The widespread applicability of Deep Generative Models (DGM) such as variational autoencoders (VAEs) has been demonstrated in many domains that can benefit from generative modeling, including NLP tasks such as language modeling (Bowman et al., 2016). DGMs for text aim to model holistic textual properties such as style and topics via encoded variables to improve performance on, e.g. word imputation and other tasks. However, a common problem of VAE models is their tendency to ignore latent variables on which they should condition. Recent work has devoted much attention to solving this problem, known as *posterior collapse* (Pelsmaeker and Aziz, 2019).

We investigate and compare the applicability of posterior collapse mitigation using Minimum Desired Rate (Pelsmaeker and Aziz, 2019) and FreeBits (Kingma et al., 2016), both separately and in combination with randomly masking words during decoding (WordDropout). Our contributions are a more in-depth understanding of the usefulness of these methods and their respective tradeoffs.

We begin the paper by briefly discussing related work in Section 2, followed by an overview of our approach in the next section. We then present the most important analytical results in Section 4, and finally a detailed discussion of the implications and future directions of our research, qualitative evaluation, and conclusion in Section 5.

2 Related Work

Previous research has focused on VAEs for text and posterior collapse methods. Bowman et al. (2016) introduced the Sentence VAE architecture for its generative properties and the ability to extract global semantic features. Their architecture uses LSTM cells for encoding and decoding, and diagonal Gaussian prior and posterior distributions. We use similar distributions, but instead of LSTM cells we use Gated Recurrent Units (GRUs) (Cho et al., 2014). The authors introduce Word Dropout and KL cost annealing techniques to prevent posterior collapse. We also experiment with the former. Kingma et al. (2016) introduce a novel VAE architecture that includes normalizing flows and more improvements such as an objective function specifically suited to mitigating posterior collapse, dubbed Free Bits. The KL-divergence term of the ELBO objective is constrained to be larger than λ per group or dimension of latent variables. The constraint is applied with a simple max function. Pelsmaeker and Aziz (2019) review existing posterior collapse mitigation methods, and introduce and review their own, Minimun Desired Rate (MDR). MDR is similar to FreeBits in the sense that it imposes a minimum constraint on the KL-divergence term of the ELBO. Contrary to FreeBits, in MDR this constraint is imposed on the sum over all dimensions of the KL-divergence, and constrained optimisation through a Lagrangian is used.

3 Approach

Dataset & models The dataset we used is the Penn Treebank (PTB) dataset (Marcus et al., 1993). This dataset consists of 43.948 American English sentences, with an approximate (91/4/5) train/validation/test split.

In a VAE architecture, a distribution over latent variables, underlying the real distribution of data, is modeled in terms of: the *prior distribution* $p(\mathbf{z})$, the *conditional likelihood* of the data $p(\mathbf{x}|\mathbf{z})$, and the *posterior distribution* of the latent variables $p(\mathbf{z}|\mathbf{x})$. The goal is to evaluate this posterior, as it defines the distribution of $\mathbf{z}$ given a datapoint.

However, this posterior distribution is intractable when using neural networks, so in a VAE an *approximate posterior* is modeled, $q(\mathbf{z}|\mathbf{x})$. The approxi-

mate posterior (encoder) and likelihood (decoder) are parameterised by neural networks, with parameters $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, respectively. Directly optimizing this posterior with respect to the marginal probability $p(\mathbf{x})$ over q is not tractable either, so instead a lower bound is computed. This *Evidence Lower Bound*, ELBO in short, is formulated according to equation 2. The first term of the ELBO can be seen as the reconstruction accuracy, which should be maximised. The second term can be seen as a regularisation term that minimises the dissimilarity between q and the prior (Kingma and Welling, 2014).

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})\right] \quad (1)$$

$$- \mathcal{D}_{KL}\left(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right) \quad (2)$$

VAEs have many benefits, including the ability to generate new samples from a latent distribution, and the ability to model continuous latent features in the dataset. Applying VAEs to language is therefore very natural, since language is full of latent information, such as style or topic.

Similarly to the work of Bowman et al. (2016), we set up a Sentence VAE as a language model, using GRU cells as encoders and decoders. The output of the encoder is transformed by two separate linear layers into mean and scale vectors, which form the parameters for the posterior distribution. One or multiple samples $\mathbf{z}$, representing the latent variable, are transformed by a linear layer to form the first hidden state for the decoder $\mathbf{h}_0$. The decoder uses a GRU-cell and a linear layer to generate a probability for each token in the vocabulary at each timestep, after which greedy sampling is used to select the output token. The sequence generation continues until the model generates an end-of-sequence token. During training, the input token for the decoder-RNN at each timestep is the ground-truth previous token. Both prior and posterior are chosen as isotropic Gaussians. During training, we estimate the expected value in the first term of the ELBO using a single sample drawn from $q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.

Word Dropout Similar to using dropout to decrease the dependence of a model on specific neurons, Word Dropout attempts to decrease the decoder model's dependence on previous words in the sentence. This rewards the model for depending more on the latent variable, so makes posterior collapse less beneficial. Words are left out and replaced by an unknown token at random during the decoding step, with a given keep rate, which is tuned as a hyperparameter. (Bowman et al., 2016).

FreeBits The FreeBits method makes use of a modified objective: the KL divergence per dimension of the latent variable is limited to a minimum value λ. This removes the advantage of a KL term that approaches zero, so that the model can keep using the latent variable for decoding. Note that this limitation is applied after averaging the KL term over the minibatch, when using minibatch gradient descent. This objective is formalised in equation 4, where $\mathbb{E}_{batch}$ indicates averaging over a (mini)batch, z_j is the j-th component of $\mathbf{z}$, and K the dimensionality of $\mathbf{z}$. (Kingma et al., 2016)

$$\tilde{\mathcal{L}}_{\lambda} = \mathbb{E}_{batch}\left[\mathbb{E}_{q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\phi}}(\mathbf{x}|\mathbf{z})\right]\right] \quad (3)$$

$$- \sum_{j=1}^{K} \max\left(\lambda, \mathbb{E}_{batch}\left[\mathcal{D}_{KL}\left(q_{\boldsymbol{\theta}}(z_j|\mathbf{x})||p(z_j)\right)\right]\right) \quad (4)$$

Minimum Desired Rate With FreeBits, the KL-term is not prohibited from going to zero, rather, the reward for the model for bringing it to zero is taken away. Additionally, there is a discontinuity in the gradient of the ELBO where $z_j = \lambda$. This constraint, as such, works in practice, but its theoretical grounding is not very strong. A better approach would be to impose a hard constraint on the KL-term, and use constrained optimisation. Minimum Desired Rate does this by applying a *minimum rate* to the total KL-term. Through optimisation using a Lagrange multiplier $u \in \mathbb{R}_+$, the ELBO is maximised with the KL-term constrained to being larger than r. The objective function, equation 5, is maximised for $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, and minimised for u, using gradient ascent and descent, respectively. (Pelsmaeker and Aziz, 2019)

$$\Phi(\boldsymbol{\theta}, \boldsymbol{\phi}, u; \mathbf{x}) =$$
$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{x}) - u(r - \mathbb{E}_{batch}\left[\mathcal{D}_{KL}\left(q_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})||p(\mathbf{z})\right)\right] \quad (5)$$

4 Experiments and Results

Language Modeling In our task, the model is asked to predict the next word in a sentence. It can use information from *previous* words in the sequence for this, and in the autoencoder setup of the VAE, also some *global information* of the

sentence. During training and testing, we determine the log-likelihood of the ground truth word at each timestep, to use in the loss function. Additionally, the ground truth previous word is fed into the model, rather than the generated previous word. During generation, the model uses the previous words it generated, until and end-of-sequence token is output.

The following metrics are used to judge the quality of the models. *Negative log-likelihood (NLL)*: This is the summed negative log-likelihood of the ground truth words at each time-step. *Kullback-Leibler (KL) Divergence*: The KL divergence is not directly a measure for the quality of the model, but it is indicative of how well posterior collapse is mitigated. And finally, *perplexity (PPL)*, based on the marginal log-likelihood, which is estimated using importance sampling.

Baseline RNN The baseline RNN uses only information from the previous words in the sentence, and is initialized with a hidden state vector of zeros. Dropout layers are used between the word embedding and RNN layer, and between the RNN layer and the final linear layer. The optimiser used is Adam, with a learning rate of 0.001 (Kingma and Ba, 2014). The model generally took 4 epochs to converge.

Vanilla Sentence VAE The Sentence VAE encoder is applied to each complete sentence, after which the decoder RNN uses the latent variable as a first hidden state. This decoder RNN is then applied to the complete sentence in a similar way to the baseline RNN. The optimiser here is Adam as well, with a learning rate of 0.001. Our hyperparameter search started at the same parameters as used in Bowman et al. (2016), mostly rounded to powers of two. Any changes we made to those hyperparameters did not improve performance. This model is expected to perform similarly to or worse than the baseline, when posterior collapse happens. The VAE generally took 4 or 5 epochs to converge.

The results show that this model does indeed perform worse than the baseline, and the KL score indicates that posterior collapse happened.

Posterior Collapse Mitigation Methods We apply the methods for posterior collapse mitigation to the vanilla Sentence VAE, keeping all hyperparameters the same. We experimented with each of the methods in isolation, and with the combination of Word Dropout and FreeBits or MDR. We did not combine FreeBits and MDR, as they are based on a similar principle.

Model	NLL (Std)	KL	PPL
RNN (baseline)	126.4 (0.30)	-	220.3 (2.84)
Vanilla VAE	127.0 (0.18)	0.055 (0.010)	224.3 (1.67)
Word Dropout	126.8 (0.42)	1.2 (0.15)	230.8 (3.29)
FreeBits	121.0 (0.15)	8.3 (0.066)	230.1 (1.25)
FreeBits*	121.5 (0.32)	8.8 (0.28)	231.1 (2.81)
MDR	119.8 (0.22)	9.7 (0.13)	238.7 (3.80)
MDR*	121.0 (0.46)	9.4 (0.36)	239.1 (4.40)

Table 1: Test performance for the considered models, measured by negative log-likelihood (**NLL**), KL-divergence (**KL**), and perplexity (**PPL**). Standard deviation over four model initialisations is included in brackets. *Combined with Word Dropout

Word Dropout We use a word dropout keep rate of 0.66, and this rate was also used for the combined experiments. The performance is slightly better than vanilla, but still not as good as the baseline. Note that the KL value is larger now, indicating that posterior collapse has happened to a lesser extent. We conclude that Word Dropout helps, but is not sufficient in isolation.

Freebits The FreeBits implementation was done by clamping the value of the KL divergence for each dimension to the specified λ value, after which this was used to compute the ELBO. For FreeBits in isolation, and in combination with Word Dropout, we found an ideal λ value of 0.5, which is in line with Kingma et al. (2016).The performance is notably better than the baseline, and the KL score is significantly higher. This shows that posterior collapse is effectively avoided. This goes for both in combination and without Word Dropout. Without Word Dropout, results are slightly better, and the KL term is slightly higher.

MDR We implemented MDR by reformulating the loss function to equation 5. We defined a second PyTorch optimiser specifically for u, and negated the gradient of this optimiser before updating all parameters. We used RMSProp for this optimiser, similar to Pelsmaeker and Aziz (2019), and found that other optimisers did not work as well. For MDR, the best desired rate r was found to be 10.0. Using MDR in isolation resulted in better performance than FreeBits. Combining it with Word Dropout results in a slightly worse performance. The KL terms were higher compared to FreeBits.

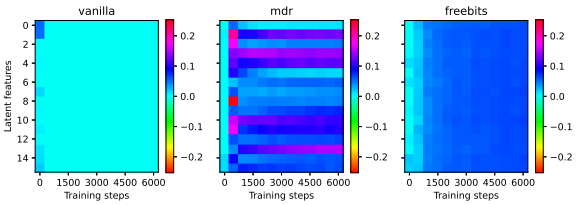

Figure 1: Progression KL Divergence per dimension of the latent variable (vertical axis) over training steps (horizontal axis).

5 Discussion

From the results in the previous section, we see that using MDR with Word Dropout is the best method for preventing posterior collapse. Combining FreeBits with Word Dropout results in only slightly worse performance. We can conclude that both of these methods work reasonably well to prevent posterior collapse, and are robust.

For both methods, we see the overall performance improving over the RNN, which indicates that the VAE is better at generalising.

Sampled Sentences In Table 2 (appendix), we show some sampled sentences. The sentences sampled from the Vanilla VAE are all identical, which indicates that it does not use the randomly sampled latent variable at all. This is a sign of posterior collapse. Interestingly, most of the Word Dropout model's sentences contain the word time, indicating that the posterior is still not very expressive. The FreeBits and MDR models both show quite varying sentences, but their coherence seems to have gone down. This indicates a weakness in the decoder RNN. Another architecture, such as an LSTM, could be better suited to this task.

KL Divergence Progression When posterior collapse does not happen, gradually, more information is encoded in the posterior. To visualise this process, we created a plot of the training KL term value, per latent dimension, over training steps, figure 1.

The left graph shows the progression of the KL term for the vanilla VAE. After few training steps, it already converges to zero, indicating a quick posterior collapse. None of the dimensions encode any information. For FreeBits (right), the KL term uniformly converges to a non-zero value. This implies that information is encoded uniformly in the latent variable.

For MDR (middle), all dimensions show a different behaviour. Some of them converge to large values, a few become zero, and some converge to intermediate values. Each dimension of the latent variable seems to encode information in a different way. Collapse still happens for some of the dimensions. The difference between MDR and FreeBits in these graphs can be attributed to their theoretical differences. FreeBits imposes a minimum *per dimension*, so each dimension encodes enough information to adhere to this minimum. MDR imposes a total limit to the KL term, so some dimensions are still allowed to collapse, as long as others become larger. Additionally, since MDR has a continuous gradient of the ELBO, it is better able to improve expressivity of each dimension through parameter updates. The FreeBits approach relies on the stochasticity of gradients for this.

Future Research In future work, we would like to study the applicability of different prior distributions. Namely, a further examination of the distributional properties of the learned latent space could give more insights as to which family of probability distributions would be more suitable as conjugate priors for a VAE architecture, and thereby be more effective in not being ignored during training. A deeper dive into finding better optimisers for MDR and expanding on the work of Pelsmaeker and Aziz (2019) would also be interesting. Finally, we would like to study a more directed enforcing of specific latent variables, such as explicit topic or style, on text generation.

Conclusion We experimented with three Posterior Collapse Mitigation methods: Word Dropout, FreeBits and MDR. We found that each of these methods decreased posterior collapse, with especially FreeBits and MDR resulting in large improvements. Out of these two, MDR is better. We also visualised and discussed the progression of the KL divergence term over training, under the application of these methods. We saw that the MDR method incites the largest variance in posterior dimensions, while FreeBits causes a very uniform spread of the posterior dimensions. Additionally, we sampled some sentences from the models, and note that the models that do not have posterior collapse show more varying sentences, that are less coherent, as compared to the models that do reach posterior collapse.

References

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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

D.P. Kingma and M. Welling. 2014. Auto-encoding variational bayes. *Conference proceedings: papers accepted to the International Conference on Learning Representations (ICLR) 2014*, (ICLR).

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. 2016. Improved variational inference with inverse autoregressive flow. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4743–4751. Curran Associates, Inc.

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

Tom Pelsmaeker and Wilker Aziz. 2019. Effective estimation of deep generative language models. *CoRR*, abs/1904.08194.

A Appendix

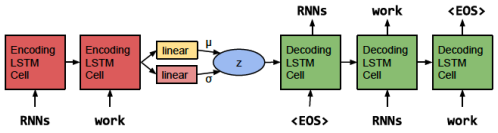

Figure 2: A graphical model of the Sentence VAE, as depicted in Bowman et al. (2016).

Vanilla	“ The market is a very good thing , ” he said “ The market is a very good thing , ” he said
Word Dropout	But the Fed has been a problem that the company ’s ability to be a good time “ I do n’t think we ’re going to be a good bet for the first time , ” said Mr. Mehl
Freebits	John F. Seib , who resigned as much as $ 1 billion in the U.S. , he said In the past two weeks , the company said it will be able to build a new series of its own account
MDR	The two men is a “ tool-and-die ” of the world ’s wife , and that ’s “ great ” “ is a duck . For the first time , and it will be able to be able to get a new line of the company ’s business

Table 2: Two sentences sampled from the Sentence VAE trained without any posterior collapse mitigation methods, with Word Dropout, with FreeBits and with MDR.

Hyperparameter \ Model	RNN	VAE
Word embedding size	512	300
Hidden state size	512	256
Number of layers	1	1
Batch size	32	32
Latent variable size	-	16

Table 3: Basic hyperparameter settings for the RNN baseline and Sentence VAE.