# NLP2 Project: *Deep Generative Models for Text*

Teaching assistant: *Eelco van der Wel*

March 28, 2020

## 1 Introduction

In recent years, there have been major advances in text generation with neural models. A strong architecture like GPT-2 [Radford et al., 2019] can produce text that is almost indistinguishable from text fragments written by human authors. However, these models have a major shortcoming: it is generally not possible to generate text conditioned on an external variable to force a specific subject, style, or sentiment. For example, in the case of GPT-2 the best we can do is priming on a piece of text, and hoping the model picks up on the specific features we want to condition on.

This is where deep generative models come in. In a probablistic setting, what we would like to do is train a joint distribution over $p(x, z) = p(z)p(x|z)$, where X is some observed sentence, and Z is an unobserved variable we infer by training the model. If we can train such a model succesfully, z captures features like subject and writing style, and we can sample from this z to generate new data or change existing text.

In this project you will implement a deep generative language model called a Sentence VAE, and investigate a problem called posterior collapse. Your tasks is to:

- Implement data pre-processing.
- Implement the models described in Section 3.
- Implement two common methods to fight posterior collapse, and investigate a more state-of-the-art posterior collapse method.
- Perform quantitative and qualitative evaluation on the test dataset.
- Write a report and present a poster.

## 2 Dataset

In this project you will use the Penn-Treebank (PTB) data set, which is used as a benchmark in many language-modelling tasks. You are free to use any preprocessing methods you need, including the code used in the Colab Tutorial in week 1.

The data sets are available on Canvas. The training set is `02-21.10way.clean`, the validation set is `22.auto.clean`, and the test set is `23.auto.clean`. Note that these data sets are not freely available, please do not upload them publicly to GitHub.

# 3 Models

## 3.1 RNN Language Model

The baseline for this project is a standard RNN Language Model (RNNLM). This model parametrizes an autoregressive distribution over words $p(X_i|x_{<i};\theta)$. In other words: We predict the distribution over words $X_i$ conditioned on the observed history $x_{<i}$, parametrized by parameters $\theta$. The full model should be as follows:

$$x_0 = [\text{BOS}] \tag{1a}$$
$$h_0 = 0 \tag{1b}$$
$$e_i = \text{Embedding}(x_i;\theta) \tag{1c}$$
$$h_i = \text{GRU}(e_i, h_{i-1};\theta) \tag{1d}$$
$$f(x_{<i};\theta) = \text{softmax}(\text{Linear}(h_i;\theta)) \tag{1e}$$

where [BOS] is a `Begining Of Sentence` token. You are free to choose the number of layers, size of the model, and regularization techniques.

## 3.2 Sentence VAE

Next, implement the Sentence VAE as described in Bowman et al. [2015]. Before you start, take a look at section 4 for some implementation pointers.

We can split the Sentence VAE into two parts: an Inference model (or, encoder) and a Generative model (decoder). In your implementation, the generative model should follow the exact specification of the RNNLM in section 3.1. However, instead of conditioning the RNNLM on the zero vector (Equation 1b), we condition on the latent variable Z:

$$h_0 = \tanh(\text{Linear}(z;\theta)) \tag{2}$$

To train this model, we need to compute the gradients for the marginal log-likelihood $\log P(x) = \log \int P(x,z)\mathrm{d}z$. This is generally intractable, so we resort to variational inference. Instead of computing the gradients exactly, we use an approximate posterior $q(z|x)$ and optimize the Evidence Lower Bound (ELBO). We define the prior and variational posterior as follows:

$$p(z) = \mathcal{N}(\mathbf{0}, \mathbf{1}) \tag{3a}$$
$$q(z|x;\phi) = \mathcal{N}(\mu(x;\phi), \sigma(x;\phi)) \tag{3b}$$

The variational posterior $q(z|x;\phi)$ is parametrized by a bidirectional RNN:

$$h_0 = 0 \tag{4a}$$
$$e_i = \text{Embedding}(x_i;\phi_{\text{emb}}) \tag{4b}$$
$$f_i = \text{GRU}(e_i, h_{i-1};\phi_f) \tag{4c}$$
$$b_i = \text{GRU}(e_i, h_{i+1};\phi_b) \tag{4d}$$
$$\mu = \text{Linear}([f_n; b_1];\phi_\mu) \tag{4e}$$
$$\sigma = \text{softplus}(\text{Linear}([f_n; b_1];\phi_\sigma)) \tag{4f}$$

Where $[\cdot;\cdot]$ is the concatenation operator. Note that this parametrization deviates slightly from the original Gaussian VAE in Kingma and Welling [2013]. The softplus activation forces a positive output, and makes training more stable by omitting the logarithm in the original Gaussian VAE specification.

### 3.3 Posterior Collapse

In your report, explain what posterior collapse is and why it happens, and implement the following methods to mitigate collapse:

- Word Dropout [Bowman et al., 2015]
- Freebits [Kingma et al., 2016] Appendix C8

Explain how these methods reduce collapse, and compare the Sentence VAE with and without these additions.

*Note*: Bowman et al. [2015] propose an additional method called KL annealing. In recent work Freebits is generally preferred over KL annealing, so we will not implement KL annealing in this project. This also means that KL annealing does not count as a possible free choice in section 3.4.

### 3.4 Posterior Collapse: Free choice

As a final step, investigate a more recent method to reduce posterior collapse. You have three options:

1. Choose a method from the list in 3.4.1;
2. Search in literature for a different method;
3. Come up with your own idea.

If you choose to go for option 2 or 3, **be sure to discuss your plans with your teaching assistant first.** Compare your chosen method to the Sentence VAE from Section 3.2, and to the two methods in Section 3.3.

#### 3.4.1 Posterior Collapse methods

The following list contains a few more recent approaches to fight posterior collapse.

- Skip-VAE, [Dieng et al., 2018] Section 4.2
- $\mu$-forcing [Liu et al., 2019] Section 4
- Independent $\delta$-VAE, [Razavi et al., 2019] Appendix D
- Minimum Desired Rate, [Pelsmaeker and Aziz, 2019] Section 4

## 4 Pytorch Pointers

Pytorch includes many distributions and methods to model stochastic nodes. I advise you to take a look at the `torch.distributions` package, the distributions included in this package have methods to produce reparametrizable samples (`Distribution.rsample`), calculate KL divergence (`torch.distributions.kl_divergence`), and much more. You are free to implement these methods yourself as well, but this is not the focus of this project.

# 5 Deliverables

1. Colab notebook, due April 24, 2020. The notebook should contain the entire pipeline from data generation to model training to the analysis conducted. Functions or classes are allowed to be defined in Python files externally, as long as the main functionality is listed in the notebook. We recommend training your models on GPUs through the Google Colab service.

2. Short paper, due April 24, 2020. The short paper should contain max four pages (references excluded). A suggested page distribution is as follows:

   (a) **Introduction**: introduce the reader to your research area, summarise your contributions and highlight the relevance of your research (0.6 pages);

   (b) **Related Work**: summarise research papers relevant for your work. Be brief, since this is a short paper (0.4 pages);

   (c) **Approach**: the content of this section depends on your project. (1 page);

   (d) **Experiments and Results**: detail the precise experimental setup used and the numerical results your models achieved (1 page);

   (e) **Discussion**: (1 page).

   For the *Sentence VAE* project, at least the following components should be included:

   (a) A short description and motivation of the Sentence VAE and Posterior collapse methods.

   (b) A description of the models you trained, along with the hyperparameters experimented with and the final experimental setup.

   (c) A comparison between the different models, with:

      i. perplexity

      ii. log-likelihood for the RNNLM, and multi-sample estimates of the log-likelihood for VAE models.

      iii. KL divergence between the variational posterior and prior.

   (d) A discussion between the different posterior collapse methods you've tested.

   (e) Include qualitative evaluations such as samples and reconstructions from the model, or interpolations between sentences. These evaluations can be included as appendix.

3. Poster Presentation, due April 22, 2020. Compress the paper's content into a single-page poster that could be presented at a conference. Support the textual content through visual aids, such as tables and graphs that facilitate fast understanding of the paper's contributions and main results.

# 6 Suggested Schedule

To stay on track, we recommend adhering to the following schedule. The lab sessions will start with a short presentation on the week's topic, containing references to the recommended reads to serve as inspiration and extend your knowledge on variational inference and deep generative language models.

## 6.1 Week 1

**Reading** To prepare for the Sentence VAE paper, read Auto-Encoding Variational Bayes [Kingma and Welling, 2013]. For a more in-depth look, you can go over the tutorial by Doersch [2016] (Optional).

**Coding** Write your pre-processing code for PTB, and implement the RNNLM.

**Writing** It is a good habit to write about the research along the way. At this point, you can already start writing an Introduction, dataset section, and a section baseline models.

## 6.2 Week 2

**Reading** Go over the Sentence VAE in Bowman et al. [2015], and appendix C8 in Kingma et al. [2016].

**Coding** Implement the Sentence VAE as described in section 3.2, and analyze the differences between the RNNLM and VAE models. Additionally, Implement Freebits and Word Dropout.

**Writing** Fill out the Experiments and Results sections of your report.

## 6.3 Week 3

**Reading** Go over the suggested papers in Section 3.4.1. You are not required to read the full papers, but at least get an understanding of the methods put forth in the sections relevant to this project.

**Coding** Implement the posterior collapse method of your choice. This is also the time to finish your evaluation code.

**Writing** Write the 'Discussion' section of the paper. Summarise the conclusions of your analyses in text and ensure that these are supported by evidence presented in tables or graphs. Create an outline for the poster to be presented.

## 6.4 Week 4

**Reading** Any papers relevant to your chosen posterior collapse method.

**Coding** Ensure that the notebook is understandable for the reviewers and that it details the entire pipeline from data generation to the analysis of results.

**Writing** Finish the paper and edit its content to fit into four pages. Additional analyses may be contained in the appendices, but the grade will be graded based on these first four pages. Include the paper's main points in the poster, supported by visual aids, such as tables and graphs that facilitate fast understanding of the artificial task, the most interesting results and the analyses conducted.

# References


Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Adji B Dieng, Yoon Kim, Alexander M Rush, and David M Blei. Avoiding latent variable collapse with generative skip models. *arXiv preprint arXiv:1807.04863*, 2018.

Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

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

Durk P Kingma, Tim Salimans, Rafal Jozefowicz, Xi Chen, Ilya Sutskever, and Max Welling. Improved variational inference with inverse autoregressive flow. In *Advances in neural information processing systems*, pages 4743–4751, 2016.

Dayiheng Liu, Yang Xue, Feng He, Yuanyuan Chen, and Jiancheng Lv. $\mu$-forcing: Training variational recurrent autoencoders for text generation. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(1):1–17, 2019.

Tom Pelsmaeker and Wilker Aziz. Effective estimation of deep generative language models. *arXiv preprint arXiv:1904.08194*, 2019.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9, 2019.

Ali Razavi, Aäron van den Oord, Ben Poole, and Oriol Vinyals. Preventing posterior collapse with delta-vaes. *arXiv preprint arXiv:1901.03416*, 2019.