# Do sequence-to-sequence VAEs learn global features of sentences?

**Tom Bosc**
Mila; Université de Montréal
bosct@mila.quebec

**Pascal Vincent**
Mila; Université de Montréal; CIFAR
vincentp@iro.umontreal.ca



## Abstract

A longstanding goal in NLP is to compute global sentence representations. Such representations would be useful for sample-efficient semi-supervised learning and controllable text generation. To learn to represent global and local information separately, Bowman et al. (2016) proposed to train a sequence-to-sequence model with the variational auto-encoder (VAE) objective. What precisely is encoded in these latent variables expected to capture global features? We measure which words benefit most from the latent information by decomposing the reconstruction loss per position in the sentence. Using this method, we see that VAEs are prone to memorizing the first words and the sentence length, drastically limiting their usefulness. To alleviate this, we propose variants based on bag-of-words assumptions and language model pretraining. These variants learn latents that are more global: they are more predictive of topic or sentiment labels, and their reconstructions are more faithful to the labels of the original documents.

## 1 Introduction

Natural language generation is a major problem underlying many classical NLP tasks such as machine translation, automatic summarization or dialogue modeling. Recent progress has been mostly attributed to the replacement of LSTMs (Hochreiter and Schmidhuber, 1997) by more powerful, attention-based models such as Transformers (Vaswani et al., 2017; Radford et al., 2019).

Despite their differences, Transformers remain mostly used in an auto-regressive manner via masking, generating words one after the other. In contrast, the sequence-to-sequence model trained with a Variational Auto-Encoder (Kingma and Welling, 2013; Rezende et al., 2014) (*VAE*) objective proposed by Bowman et al. (2016) generates text in a two-step process: first, a latent vector is sampled from a prior distribution; then, words are sampled from the probability distribution produced by the auto-regressive decoder, itself conditioned on the latent vector. The hope is that such an architecture would encourage a useful information decomposition, where the latent vector would "explicitly model holistic properties of sentences such as style, topic, and high-level syntactic features", while the local and grammatical correlations would be handled by the recurrent decoder. Such global features encoded in a compact, fixed-size representation would be handy both for semi-supervised learning and controllable generation. For semi-supervised learning, the latent vector would be the ideal representation on which to train small classifiers using a handful of labels (Kingma et al., 2014). For controllable generation, we could obtain a "prototypical" latent vector for a given label by averaging the latent vectors of all datapoints sharing that label. Then, we could decode this average vector to generate examples that would be labeled similarly.

Despite its conceptual appeal, Bowman et al. (2016)'s VAE suffer from the *posterior collapse* problem. The VAE objective is a sum of two terms: a reconstruction term that encourages the encoder and decoder to collaborate to reconstruct the input, and a KL divergence term that aligns the approximate posterior produced by the encoder with the prior. The problem is that, early on during training, the KL term goes to 0, such that the approximate posterior becomes the prior and no information is encoded in the latent variable. Faced with the same problem in the context of image modelling, Chen et al. (2016) remarked[1] that it is possible to imagine a model architecture where learned latent variables would encode local statistics while the auto-regressive decoder would focus on global variations. In summary, latent variables can be com-

[1] In Section 3.1.

pletely uninformative or encode local information in an undesirable and counter-intuitive manner.

Using modifications to the objective such as free bits (Kingma et al., 2016), we can obtain a positive KL term, which indicates that *some* information is encoded in the latent variables. However, how can we verify that they capture *global* aspects of texts? Qualitative evaluation methods such as reconstruction from interpolated codes ("homotopies") are highly subjective and ill-defined. Semi-supervised experiments are useful, but as we will show, they are limited and often not performed correctly.

In this paper, we propose to examine the content of latent variables by decomposing the reconstruction loss over positions in the sentence. We observe that encoders mostly store in the latent vector information pertaining to the first few words of each sentence as well as the number of words. If sequence-to-sequence VAEs sometimes encode global features, it is a byproduct of this memorization behavior, and therefore depends heavily on the dataset. This casts serious doubts about the usefulness and robustness of these representations. To prevent this behavior, we propose simple variants based on bag-of-words assumptions and pretraining. The representations learned by our variants are more predictive of the ground-truth labels, both in the small or large data-regime. Consequently, the reconstructions of texts share the same label as the source texts more often than our baselines and memorization is decreased.

## 2 Model and datasets

### 2.1 Sequence-to-sequence model and VAE objective

We briefly describe the object of this study, the sequence-to-sequence model with the VAE objective (Bowman et al., 2016). A document, sentence or paragraph, of $L$ words $x = (x_1, \dots, x_L)$ is embedded in $L$ vectors $(e_1, \dots, e_L)$. An LSTM encoder processes these embeddings to produce hidden states:

$$h_1, \dots, h_L = \text{LSTM}(e_1, \dots, e_L)$$

In general, the encoder produces a vector $r$ that represent the entire document. In the original model, this vector is the hidden state of the last word $r = h_L$, but we introduce variants later on. This representation is transformed by linear functions $L_1$ and $L_2$, yielding the variational parameters that are specific to each input document:

$$\mu = L_1 r$$
$$\sigma^2 = \exp(L_2 r)$$

These two vectors of dimension $d$ fully determine the approximate posterior, a multivariate normal with a diagonal covariance matrix, $q_\phi(z|x) = \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$, where $\phi$ is the set of all encoder parameters (the parameters of the LSTM, $L_1$ and $L_2$). Then, a sample $z$ is drawn from the approximate posterior and the decoder, another LSTM, produces a sequence of hidden states:

$$h'_1, \dots, h'_L = \text{LSTM}([e_{\text{BOS}}; z], [e_1; z], \dots, e_L; z])$$

where BOS is a special token indicating the beginning of the sentence and $[\cdot;\cdot]$ denotes the concatenation of vectors. Finally, each hidden state at position $i$ is transformed to produce a probability distribution of the word at position $i+1$:

$$p_\theta(x_{i+1}|x_{1,\dots,i}, z) = \text{softmax}(W h'_i + b)$$

where $\text{softmax}(v_i) = e^{v_i} / \sum_j e^{v_j}$ and $\theta$ is the set of parameters of the decoder (the parameters of the LSTM decoder, $W$ and $b$). The vocabulary is augmented with an EOS token indicating the end of the sentence, which is appended at the end of every document.

For each document $x$, the lower-bound on the marginal log-likelihood (*ELBo*) is:

$$\begin{aligned}\log p(x) &\geq -D_{\text{KL}}(q_\phi(z|x)||p(z)) + \log p_\theta(x|z) \\ &\geq \text{ELBo}(x, \phi, \theta)\end{aligned}$$

On the entire training set $\{x^{(1)}, ., x^{(N)}\}$, the objective is:

$$\arg\max_{\phi,\theta} \sum_{j=1}^{N} \text{ELBo}(x^{(j)}, \phi, \theta)$$

### 2.2 Controlling the capacity of the encoder

Following Alemi et al. (2018), we call the average value of the KL term the *rate*. It measures how much information is encoded on average about the datapoint $x$ by the approximate posterior $q(z|x)$.

The KL term can be modified to target a specific rate, or at least to make sure it is above a target rate using a variety of similar techniques (see Appendix A.1 for more details). The main goal of these modifications is to prevent the posterior collapse in sequence-to-sequence VAE. We use the

free bits formulation of the $\delta$-VAE (Razavi et al., 2019): for a desired rate $\lambda$, the modified negative ELBo is:

$$\max(D_{\mathrm{KL}}(q_\phi(z|x)||p(z)), \lambda) - \log p_\theta(x|z)$$

Since sequence-to-sequence VAEs are prone to posterior collapse, in practice, the rates obtained are very close to the target rates $\lambda$.

As observed by Alemi et al. (2018), different models or sets of hyperparameters for a given model can yield very similar values of ELBos despite reaching very different rates. In other words, the work of modelling stochasticity can be divided very differently between the latent variable and the auto-regressive decoder. Therefore, for our purposes, the free-bits modification has the additional advantage that it enables us to compare different models with similar capacity.

### 2.3 Variants

Throughout the paper, we use variants of the original architecture and training procedure. In the next section, we also use a deterministic Auto-Encoder (*AE*) trained only with the reconstruction loss, as well as several other variants recently introduced to alleviate the posterior collapse.

Li et al. (2019) proposed to pretrain an *AE*, then to reinitialize the weights of the decoder and finally, to train the entire model again end-to-end with the VAE objective. The sentence representation is still the last hidden state of the LSTM encoder and therefore, we call this model and training procedure *last-PreAE*.

In the second variant, proposed by Long et al. (2019), the representation of the document $r$ is the component-wise maximum over hidden states $h_i$, i.e. $r^j = \max_i h_i^j$. We call this model *max*. In later experiments, we also consider a hybrid of the two techniques, *max-PreAE*.

We make slight, beneficial modifications to these two methods. We remove KL annealing which is not only redundant with the free bits technique but also increases the rate erratically. Moreover, we use $\delta$-VAE-style free bits techniques to achieve a rate closer to the target rate. These modifications are justified in Appendix A. Therefore, all of the models in the paper use $\delta$-VAE-style free bits without KL annealing.

### 2.4 Datasets

We train VAEs on four small versions of AGNews, Amazon, Yahoo and Yelp from Zhang et al. (2015).

| Dataset | Splits size | Label | $\lvert\mathcal{Y}\rvert$ | $H[Y]$ | NLL |
|---|---|---|---|---|---|
| AGNews | 110/10/10 | Topic | 4 | 1.39 | $128.77 \pm 0.21$ |
| Amazon | 100/10/10 | Sent. | 5 | 1.61 | $82.90 \pm 0.10$ |
| Yahoo | 100/10/10 | Topic | 10 | 2.30 | $81.91 \pm 0.36$ |
| Yelp | 100/10/10 | Sent. | 2 | 0.67 | $34.60 \pm 0.28$ |

Table 1: Datasets characteristics. $|\mathcal{Y}|$: number of different labels. $H[Y]$: entropy of labels. NLL: mean negative log-likelihood of LSTM baseline models (std. over 3 runs). Splits size: train/valid/test sizes in thousands.

Each document is written in English and consists of one or several sentences. Each document is manually labeled according to its main topic or the sentiment it expresses, and the labels are close to uniformly balanced over all the dataset. For faster training, we use smaller datasets. Characteristics of these datasets are detailed in Table 1.

## 3 Encoders prioritize information about the first words and sentence length

The ELBo objective trades off the KL term against the reconstruction term. To minimize the objective, it is worth increasing the KL term *only* if the reconstruction term is decreased by the same amount or more. With free bits, we allow the encoder to store information up to a certain extent without paying any cost. The optimisation objective becomes to minimize the reconstruction cost by using this "free" storage as efficiently as possible.

In order to visualize *what* information is stored in the latents, our method is to look at *where* gains are seen in the reconstruction loss. Since the loss is a sum over documents and positions in these documents, these gains could be concentrated: i) on certain documents, for example, on large documents or documents containing rarer words; ii) at certain positions in the sentence, for example in the beginning or in the middle of the sentence. We investigate the latter possibility.

### 3.1 Visualizing the reconstruction loss

Concretely, we compare the reconstruction loss of different models at specific positions in the sentence. The baseline is a LSTM trained with a language model objective (*LSTM-LM*). It has the same size as the decoders of the auto-encoder models.[2] Since the posterior collapse makes VAEs behave exactly like the *LSTM-LM*, the reconstruction losses between the VAEs and the *LSTM-LM* are directly

[2]Slightly smaller, because in VAEs, the inputs of the decoder are concatenated with the latent variable.

comparable. Additionally, the deterministic *AE* gives us the reconstruction error that is reachable with a latent space constrained only by its dimension $d$, but not by any target rate $\lambda$ (equivalent to an infinite target rate).[3]

On Figure 1, the left-side plot shows the reconstruction losses of different models and different target rates $\lambda$ on the Yelp dataset. As expected, for all models, raising the target rate lowers the reconstruction cost. In the extreme, *AE* obtains the lowest reconstruction loss. What is remarkable is that these gains are very focused around the beginning and the end of the sentence. To see that more clearly, we compute the *relative improvement* in reconstruction with respect to the baseline (right-hand side of Figure 1) as follows:

$$\tilde{r}(i) = \frac{\max(r_{\text{LSTM}}(i) - r(i), 0)}{r_{\text{LSTM}}(i)}$$

where $r_{\text{LSTM}}(i)$ is the reconstruction loss of the baseline.

All the models reconstruct the first couple of words and the penultimate token better than the *LSTM-LM*. In the Yelp dataset, the penultimate token is a punctuation mark which is always followed by the end-of-sentence token, and therefore, accurately predicting when this token occurs is equivalent to predicting the sentence length. Thus, we conclude that the latent variables encodes information about the sentence length. On the three other datasets, we see similar peaks on relative improvements in the beginning and the end of sentences (see Appendix). On Yelp, the situation is even worse than on other datasets: between positions 4 and 13, there is no relative improvement when $\lambda = 2$, indicating that the latent vector does not encode any global information.

If words in a document were pairwise independent, any improvement in reconstruction at a certain position would indicate that information about the word in that position were encoded in the latent variable. However, words are far from being independent, so how can we trace back the information to the encoder? First, any latent information related to the first word should not yield any improvements on the prediction of the second word, because the decoder is recurrent and trained using teacher forcing, i.e. conditioned on the true first word, so that information would be redundant. However, information related to the second word in the latent variable can help the decoder predict the first word. Therefore, improvements of the reconstruction loss in position $i$ can only be attributed to stored information pertaining to the words in positions $\geq i$. Second, the correlation between words in two positions decreases as the distance between these words grow. In effect, information pertaining to the second word yields more gains on the second word than on the first word. From these two facts, we conclude that gains for a position $i$ mostly comes from information about the word in position $i$ itself.

### 3.2 Impact during decoding

To study the concrete impact of this observation for generation, we encode and decode test documents using the *last-PreAE* variant.[4] Then, we compute the ratio of documents for which the first word in the sources and in the reconstructions match and similarly, how often the sources and their reconstructions have the same number of words. We compare these with scores obtained by a baseline model that outputs the most frequent first word given the label and the most common document length given the label. This baseline mimicks the behavior of a hypothetical VAE which would encode the labels of the documents (topic or sentiment) perfectly and nothing more.

Results in Table 2 show that with the *last-PreAE* the first words are reconstructed with much higher accuracy than if the latent vector only encoded the label. On the last two datasets, it recovers the first words on more than half of the documents whereas the baseline only recovers the first words between 12.9 and 14.1% of the time. Accurate encoding of the number of words seems less systematic than the encoding of the first few words. For example, on AGNews, the sentence length is recovered less often than our baselines. The encoding of the sentence length is more pronounced on datasets with small documents like Yahoo and Yelp.

### 3.3 Is it an issue?

To sum up, our first experiment shows that, compared to an unconditional *LSTM-LM*, the sequence-to-sequence VAEs incur a much lower reconstruction loss on the first tokens and towards the end of the sentence. Our second experiment indicates that

[3] In practice, we use the same code that uses sampling but $\log(\sigma^2)$ is not constrained in the objective by the KL term anymore.

[4] $\lambda = 8$, $d = 16$, decoding with beam search (beam of size 5).

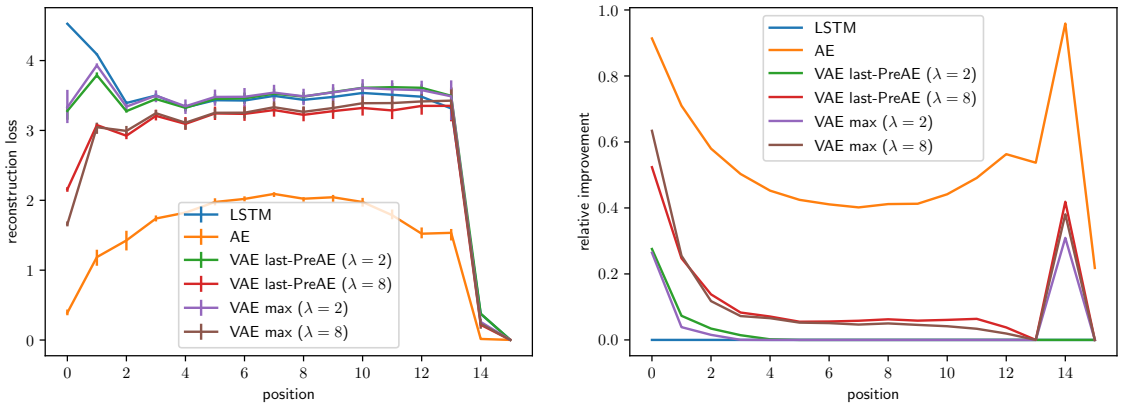


Figure 1: Reconstruction loss as a function of word position on the Yelp dataset. *Left*: reconstruction loss for each position in the sentence, averaged on sentences of 15 words and with 3 different seeds (error bars indicate min and max average of 3 runs); *Right*: relative improvement of each model compared to the baseline LSTM. Auto-encoders (vanilla and variational) consistently store information about the first couple of words as well as the sentence length.

| Dataset | *last-PreAE* 1st (%) | *last-PreAE* Len. (%) | Clf. given label 1st (%) | Clf. given label Len. (%) |
|---|---|---|---|---|
| AGNews | $29.6 \pm 1.1$ | $3.6 \pm 0.1$ | 12.9 | 4.8 |
| Amazon | $42.4 \pm 2.3$ | $13.0 \pm 1.6$ | 14.0 | 0 |
| Yahoo | $56.6 \pm 1.0$ | $17.1 \pm 1.1$ | 11.3 | 4.9 |
| Yelp | $53.0 \pm 0.5$ | $33.7 \pm 1.7$ | 14.1 | 9.7 |

Table 2: The latent variables encode more information than the label alone, in particular, information that allows to retrieve the first word and the document length with high accuracy.

if the latent variable of the VAEs did encode the label perfectly and exclusively, they would reconstruct the first words or recover the length of each document with much lower accuracy than what is observed. Therefore, we conclude that sequence-to-sequence VAEs are biased towards memorizing the first few words and the sentence length.

However, Figure 1 also shows that when enough capacity is given as free bits ($\lambda = 8$), there are consistent gains of around 0.2 nats on average in intermediary positions. In that case, we cannot claim that the encoded information is purely local. Since we can increase the capacity via the hyperparameters, is it a real issue? We believe it is for the following reasons.

Firstly, as noted by Alemi et al. (2018), higher KL values lead to lower ELBos or marginal likelihoods. Prokhorov et al. (2019) confirmed that models with low likelihood are also poor at generation and samples are less and less coherent as the rate increase. Moreover, decoding interpolations of two latent codes yield completely unrelated texts. It is often argued that more complex prior or approximate posterior are the solutions to such "non-smooth" latent spaces, but Pelsmaeker and Aziz (2019) did not find that such methods reach higher rates without any loss in the likelihood. These papers all support the idea that with current techniques, higher rates come at the cost of worse modelling of the data. Therefore, for our purposes, we should strive for latent-variable models which store less information, but more global information.

Secondly, we see potential issues related to specific use cases of VAEs. For controllable generation, we want to generate a variety of sentences that include different lengths and beginnings for a fixed, global aspect such as topic or sentiment. It is an undesirable side-effect that the particular choice of the first word or the sentence length are so strongly influenced by the latent variable. In some applications, it might be useful to learn a decoder that learns to continue a given "prompt" (the beginning of a text), but left-to-right models such as GPT-2 (Radford et al., 2019) are naturally more fit for this task. As for semi-supervised learning using such representations, downstream classifiers risk picking up on correlations that might exist between the first words or sentence lengths and the label, yielding classifiers that are not very robust or simply inefficient.

If this reasoning is correct (which we will verify in later sections), it is doubtful that the commonly used sequence-to-sequence VAE architectures in

the low capacity regime would learn a useful representation. This brings us to the third problem: most of the KL values reported in the literature are low[5]. Therefore, it is not clear whether the gains in performance (however measured) of these VAE models are significant, and if they are, what precisely cause these gains.

## 4 Proposed models

What architectures could avoid the memorization phenomenon that we have exposed? We investigate simple variants and refer to the Appendix D.1 for a more thorough comparison with existing models.

Our first variant has a simple bag-of-words (*BoW*) encoder in place of the LSTM encoder and the sentence representation $r^j = \max_i e_i^j$ where the exponents denote components and the indices denote positions in the sentence. We call it *BoW-max-LSTM*. It is similar to the max-pooling model of Long et al. (2019) except that the maximum is taken over (non-contextualized) embeddings rather than LSTM hidden states. As Long et al. (2019) reported, the max-pooling operator is better than the average operator, both when the encoder is a LSTM and *BoW*. It is possibly because the maximum introduces a non-linearity, unlike the average. Therefore, we use the maximum in all our subsequent experiments. A priori, we think that since word order is not provided to the encoder, the encoder will be unable to learn to store information pertaining specifically to the first words.

For our second variant we use a unigram decoder (*Uni*) in place of an LSTM decoder. It produces a single output probability distribution for all positions in the sentence $i$, conditioned only on the latent variable $z$. This distribution is obtained by applying a one-hidden layer MLP followed by softmax to the latent vector: $p_\theta(x_i|z) = \mathrm{softmax}(W_2\mathrm{ReLU}(W_1 z) + b)$, where $\mathrm{ReLU}(x) = \max(x, 0)$ is applied component-wise. Since the decoder does not model the order of the words anymore, we hope that the encoder will learn representations that do not focus on the reconstruction of the first words. We can use any encoder in combination of this decoder and notably, if we use a *BoW* encoder, we obtain the NVDM model of Miao et al. (2016).

Both the *BoW* encoders and *Uni* decoders variants might benefit from the *PreAE* pretraining technique, which is orthogonal. Since it is neither well understood nor well motivated (it is "a surprisingly effect fix") and would require running many more experiments, we leave it for future work.

Lastly, the pretrained LM (*PreLM*) variant is obtained in two training steps. First, we pretrain a *LSTM-LM* on each entire dataset. Then, it is used as an encoder without further training, so that the effect of pretraining can not be overridden. We use average pooling over the hidden states to get a sentence representation, i.e. $r = \frac{1}{L}\sum_{i=1}^{L} h_i$, and learn the transformations $L_1$ and $L_2$ that compute the variational parameters. Initially, we have tried to use max-pooling but the training was extremely unstable. The LM objective requires the hidden state to capture both close correlations between words but also more global information to predict long-distance correlations. The hope is that this global information information can be retrieved via pooling and encoded in the variational parameters. The *PreLM* variant is therefore nothing more than the use of a pretrained LM as a feature extractor (Peters et al., 2018). To our knowledge, this approach has not yet been evaluated in the VAE setting. Our goal here is to test the effect of the training procedure rather than the architecture, which is why we keep a simple LSTM instead of more powerful architecture such as Transformers.

We rename the baselines according to our changes, for instance, we call Li et al. (2019)'s model *LSTM-last-LSTM-PreAE*. These variants allow us to isolate the influence of the encoder, the decoder and the training procedure on the performance of the VAE.

## 5 Semi-supervised learning evaluation

We diagnosed a potential problem with sequence-to-sequence VAEs and proposed several alternative models and a training procedure to solve them. For our first evaluation, we simulate the semi-supervised learning (*SSL*) setting to see which variants produce the most informative representations. There are two training phases: first, an unsupervised pretraining phase where VAEs are trained; second, a supervised learning phase where classifiers are trained to predict ground-truth labels given the latent vectors encoded with the encoders

[5]Most papers do not report if they use bits or nats (1 bit is $\ln(2) \approx 0.693$ nats). At the risk of over-estimating their reported rates, we assume nats. Here are some of the KL values of the best models in several papers (datasets between brackets): Bowman et al. (2015): 2.0 (PTB) ; Long et al. (2019): 3.7 (Yahoo), 3.1 (Yelp); Li et al. (2019): 15.02 (Yahoo), 8.15 (PTB); He et al. (2019): 5.6 (Yahoo), 3.4 (Yelp); Fu et al. (2019): 1.955 (PTB), etc.

of the VAEs. This is essentially the same setup as *M1* from Kingma et al. (2014). We could integrate the labels into the generative model as a random variable that is either observed or missing in order to obtain better results (Kingma et al., 2014), but our goal is to study the inductive bias of the sequence-to-sequence VAE *as an unsupervised learning method*. The small and the large data-regimes give us complementary information. Informally, with many labels and complex classifiers, we quantify *how much* of the information pertaining to the labels is contained in the latent vector, whereas with a few labels and simple classifiers, we quantify *how accessible* this information is.

### 5.1 Model selection

For each dataset, we subsample $g = 5$ balanced labeled datasets for each different data-regimes, containing 5, 50, 500 and 5000 examples per class. These labeled datasets are used for training and validating during the supervised learning phase. The performance of the classifiers are measured by the macro F1-score on the entire test sets.

For a given dataset in a given data-regime, we want a measure of the performance of our models that abstracts away from i) hyperparameters for the VAEs, ii) hyperparameters for the downstream task classifiers, iii) subsampling of the dataset and iv) parameter initialisation of the VAEs. As is usually done by practitioners, we optimize over the hyperparameters of the VAEs and the classifiers, eliminating i) and ii) as sources of variance. The choice of the subsample and the initialisation of the model are used to quantify the robustness of the different algorithms.

On a given dataset and in a given data-regime, for a given model, we note $F_{ij}^{H_M,H_C}$ the F1-score obtained on the test set on the subsample using seed $i$, the parameter initialisation using seed $j$, VAE hyperparameters $H_M$ and classifier hyperparameters $H_C$. We use repeated stratified K-fold cross-validation (Moss et al., 2018) to compute a validation error $\widehat{F_{ij}^{H_M,H_C}}$. For all training folds, we train logistic regression classifiers with $L_2$ regularisation and a grid-search on $H_C \in \{0.01, 0.1, 1, 10, 100\}$. We select the best classifier hyperparameter:

$$H_C^* = \arg\max_{H_C} \widehat{F_{ij}^{H_M,H_C}}$$

Then, the best VAE hyperparameter is chosen by averaging over the $s = 3$ random seeds and picking the best classifier hyperparameter,

$$H_M^* = \arg\max_{H_M} \frac{1}{s} \sum_{i=1}^{s} \widehat{F_{ij}^{H_M,H_C^*}}$$

Having optimised the hyperparameters, we compute the test set F1-score:

$$F_{ij} = F_{ij}^{H_M^*,H_C^*}$$

We report $\bar{F}_{..}$, the empirical average F1-score over $i$ and $j$. We also decompose the variance coming from the parameter initialisation and the subsampling. Note $\bar{F}_{.j}$ the empirical average F1-score for a given $j$. We report the two following quantities:

- $\sigma_{\text{init}} = (\frac{1}{s-1} \sum_{j=1}^{s} g(\bar{F}_{.j} - \bar{F}_{..})^2)^{\frac{1}{2}}$, which quantifies the variability due to the initialisation of the model ($s = 3$ different seeds),
- $\sigma = (\frac{1}{g} \sum_{i=1}^{g} \frac{1}{s-1} \sum_{j=1}^{s} (F_{ij} - \bar{F}_{.j})^2)^{\frac{1}{2}}$, which quantifies the remaining variability ($g = 5$ seeds).

In the context of ANOVA with a linear model and a single factor, these quantities are the square roots of $MS_T$ and $MS_E$ (see Appendix E).

Finally, we also add a data-regime where the entire labeled training set is used in the supervised learning phase. In that setting, we use more expressive one-hidden-layer MLP classifiers, with early stopping on a validation set. We optimise only over the hyperparameters of the VAE. This allows us to to check that our conclusions do not depend too much on the model selection procedure and on the choice of the classifier.

### 5.2 Hyperparameter sweep

For each class of model, we perform a grid search over target rates $\lambda \in \{2, 8\}$ and sizes of latent vector $d \in \{4, 16\}$.

The target rates $\lambda$ are chosen to be higher than the entropy of the labels of the documents (Table 1) as we assume that the latent variable should at least capture the annotated label. Indeed, $\lambda = 2$ nats is enough to store the labels of all datasets without any loss, except Yahoo which has an entropy of 2.3 whereas $\lambda = 8$ nats suffices to capture much more information than needed to store the labels on all datasets. Moreover, these rates are chosen to be much smaller than the reconstruction loss

of the baselines because of the technical difficulty of increasing the rate without degrading the log-likelihood explained above.

The latent vector dimension $d$ is either 4 or 16. Recall that our representations are evaluated on downstream tasks with very limited data in some cases (as little as 5 examples per class), so we need a small enough dimension of latent vector to be able to learn. We suppose that $d = 4$ will be favored for the 5 or 50 examples per class regime while $d = 16$ could be more efficient above this, but we leave this choice to the model selection procedure.

Other training details and hyperparameters kept constant are described in Appendix C.

### 5.3 What is the representation of a document?

VAEs are mostly used for generating samples but are also sometimes used as feature extractors for SSL. In the latter case, it is not clear what the representation of a datapoint is: the mean of the approximate posterior $\mu$ or the noisy samples $Z \sim \mathcal{N}(\mu, I\sigma^2)$? Kingma et al. (2014) feed noisy samples $z$ in the classifiers but in the literature of VAEs applied to language modeling, it is more common to use $\mu$ without explanation or even mention.[6]

If we are interested purely in downstream task performance, the mean should perform best, as the samples are just noisy versions of the mean vector (it is still not completely straightforward as the noise could play a regularizing role). However, in order to evaluate what information is *effectively* transmitted to the decoder, we should use the samples. The performance of downstream task classifiers using the mean does not tell us *at all* whether the latent variable is used by the decoder to reconstruct the input. The following experiment illustrates this fact.

We train the original VAE architecture on the Yelp dataset, both with and without the *PreAE*, using the original ELBo objective ($\lambda = 0$). As expected, the KL term collapses to 0. Then, we train a classifier using the procedure explained above using 5000 examples per class. We expect that its performance will be close to random chance, regardless of whether samples or the mean parameter are used as inputs. However, Table 3 shows that this is not the case. Using samples, we do get random chance predictions from the classifiers, whereas using means, the performance is remarkably high (as high as 81.5 of F1 using pretraining). The reason is that the KL term never *completely* collapses to 0. Therefore, $\mu$ can be almost zero while still encoding a lot of information about its inputs. However, when the KL term is close to 0, the variance of the samples is close to 1, so no information is transmitted to the decoder. This tendency is exacerbated with the *PreAE* runs, for which the means encode remnants of the pretraining phase.

| PreAE | F1 | | KL |
|---|---|---|---|
| | $z$ | $\mu$ | |
| No | 49.5 | 64.7 | $1e^{-4}$ |
| Yes | 49.6 | 81.5 | $2e^{-4}$ |

Table 3: When the KL collapses, the performances of downstream task classifiers trained on the mean $\mu$ vs on samples $z \sim \mathcal{N}(\mu, I\sigma^2)$ are very different, especially for pretrained models. $z$ does not contain any information while $\mu$ is very predictive of the label.

This experiment shows that it is crucial to report what representation ($z$ or $\mu$) is analyzed and to cautiously interpret the results. Therefore, for the purpose of analysing representations for text generation, we feed $z$ as inputs to the classifiers.

### 5.4 Results

Table 4 contains the results of the SSL experiments. The proposed variants are either on par or improve significantly over the baselines. In the large data-regime, *BoW-max-LSTM* and *LSTM-avg-LSTM-PreLM* perform best on average. In the small data-regime, the picture is more complex and it depends on the dataset. The exception is *LSTM-last-Uni* which is worse than the *PreAE* baselines and suffers from unstable training on AGnews (high variance).

#### 5.4.1 On which datasets do the variants improve?

On AGnews and Yelp and in the large data-regime, our variants do not seem to improve over the baselines. However, on Amazon and Yahoo, in the large data-regime, the variants seem to improve by 5 in F1-score. Why do the gains vary so widely depending on the datasets? We suppose that on some datasets, the first words are enough to predict the

[6] For instance, Li et al. (2019) and Fu et al. (2019) do not mention what representation they use but their code uses the mean; Long et al. (2019) report using a concatenation of the mean and the variance vectors.

| | Enc. | $r$ | Dec. | $n$/class Pre. | 5 | 50 | 500 | 5000 | All |
|---|---|---|---|---|---|---|---|---|---|
| | | | | | | | F1$\pm^{\sigma}_{\sigma_{\text{init}}}$ | | |
| AGNews | LSTM | last | LSTM | - | $59.6\pm^{5.1}_{11.9}$ | $71.7\pm^{1.0}_{12.1}$ | $73.6\pm^{0.1}_{11.8}$ | $73.7\pm^{0.1}_{11.9}$ | $73.6\pm^{-}_{5.4}$ |
| | LSTM | last | LSTM | AE | $65.8\pm^{3.3}_{3.3}$ | $81.0\pm^{0.7}_{1.1}$ | $82.8\pm^{0.3}_{0.6}$ | $83.1\pm^{0.1}_{0.7}$ | $83.4\pm^{-}_{0.3}$ |
| | LSTM | max | LSTM | - | $27.3\pm^{2.4}_{1.2}$ | $30.8\pm^{3.4}_{5.4}$ | $33.1\pm^{0.9}_{10.5}$ | $33.8\pm^{0.4}_{8.6}$ | $34.6\pm^{-}_{2.4}$ |
| | LSTM | max | LSTM | AE | $55.7\pm^{4.5}_{18.7}$ | $75.1\pm^{1.3}_{2.6}$ | $81.9\pm^{0.3}_{0.0}$ | $82.5\pm^{0.1}_{0.4}$ | $83.3\pm^{-}_{0.4}$ |
| | BoW | max | LSTM | - | $72.7\pm^{2.0}_{5.9}$ | $81.2\pm^{0.6}_{0.8}$ | $82.2\pm^{0.2}_{0.8}$ | $82.3\pm^{0.1}_{1.0}$ | $83.1\pm^{-}_{0.3}$ |
| | LSTM | max | Uni | - | $71.6\pm^{5.5}_{0.1}$ | $80.4\pm^{0.8}_{0.7}$ | $81.8\pm^{0.5}_{0.5}$ | $82.4\pm^{0.1}_{0.4}$ | $83.9\pm^{-}_{0.3}$ |
| | LSTM | last | Uni | - | $54.8\pm^{5.2}_{57.1}$ | $61.7\pm^{0.8}_{71.4}$ | $62.9\pm^{0.4}_{71.0}$ | $63.0\pm^{0.3}_{71.1}$ | $59.3\pm^{-}_{40.9}$ |
| | BoW | max | Uni | - | $71.8\pm^{4.5}_{1.8}$ | $81.4\pm^{0.5}_{0.6}$ | $82.5\pm^{0.1}_{0.5}$ | $82.5\pm^{0.1}_{0.6}$ | $83.1\pm^{-}_{0.5}$ |
| | LSTM | avg | LSTM | LM | $70.8\pm^{4.8}_{4.3}$ | $81.2\pm^{0.9}_{1.2}$ | $82.6\pm^{0.2}_{1.3}$ | $82.8\pm^{0.1}_{0.9}$ | $83.5\pm^{-}_{0.1}$ |
| Amazon | LSTM | last | LSTM | - | $18.9\pm^{1.7}_{0.5}$ | $20.9\pm^{1.2}_{0.9}$ | $22.5\pm^{0.7}_{0.7}$ | $23.3\pm^{0.4}_{1.1}$ | $22.9\pm^{-}_{1.5}$ |
| | LSTM | last | LSTM | AE | $20.0\pm^{2.2}_{0.9}$ | $24.7\pm^{0.7}_{2.8}$ | $27.2\pm^{0.4}_{3.1}$ | $27.7\pm^{0.3}_{3.8}$ | $28.1\pm^{-}_{1.0}$ |
| | LSTM | max | LSTM | - | $19.8\pm^{0.7}_{0.5}$ | $20.4\pm^{1.1}_{0.9}$ | $22.2\pm^{0.6}_{2.1}$ | $23.0\pm^{0.3}_{1.9}$ | $23.7\pm^{-}_{0.5}$ |
| | LSTM | max | LSTM | AE | $22.3\pm^{2.6}_{0.7}$ | $30.5\pm^{0.9}_{3.0}$ | $33.4\pm^{0.4}_{4.1}$ | $34.1\pm^{0.3}_{4.8}$ | $34.0\pm^{-}_{1.6}$ |
| | BoW | max | LSTM | - | $21.0\pm^{2.6}_{1.1}$ | $34.6\pm^{0.7}_{1.1}$ | $38.3\pm^{0.4}_{1.0}$ | $39.0\pm^{0.1}_{0.6}$ | $38.9\pm^{-}_{0.7}$ |
| | LSTM | max | Uni | - | $21.8\pm^{3.1}_{1.6}$ | $32.8\pm^{0.8}_{1.7}$ | $36.9\pm^{0.4}_{0.9}$ | $38.0\pm^{0.2}_{0.6}$ | $38.2\pm^{-}_{0.5}$ |
| | LSTM | last | Uni | - | $24.0\pm^{3.0}_{1.0}$ | $31.2\pm^{0.6}_{1.4}$ | $35.1\pm^{0.4}_{2.2}$ | $36.1\pm^{0.2}_{2.4}$ | $36.8\pm^{-}_{0.9}$ |
| | BoW | max | Uni | - | $25.4\pm^{3.2}_{0.2}$ | $32.8\pm^{1.0}_{1.3}$ | $36.1\pm^{0.4}_{0.7}$ | $36.9\pm^{0.2}_{0.8}$ | $37.9\pm^{-}_{0.2}$ |
| | LSTM | avg | LSTM | LM | $21.8\pm^{3.8}_{0.6}$ | $35.3\pm^{0.8}_{0.4}$ | $40.2\pm^{0.4}_{0.4}$ | $41.1\pm^{0.2}_{0.4}$ | $40.0\pm^{-}_{0.4}$ |
| Yahoo | LSTM | last | LSTM | - | $10.9\pm^{0.9}_{0.5}$ | $12.1\pm^{0.6}_{0.6}$ | $13.9\pm^{0.4}_{2.1}$ | $14.1\pm^{0.2}_{2.8}$ | $14.9\pm^{-}_{1.0}$ |
| | LSTM | last | LSTM | AE | $20.7\pm^{0.7}_{0.5}$ | $32.2\pm^{0.8}_{0.6}$ | $36.1\pm^{0.2}_{0.1}$ | $36.7\pm^{0.1}_{0.5}$ | $37.2\pm^{-}_{0.7}$ |
| | LSTM | max | LSTM | - | $9.9\pm^{1.0}_{1.3}$ | $13.0\pm^{0.6}_{2.1}$ | $14.6\pm^{0.3}_{2.8}$ | $14.9\pm^{0.1}_{3.1}$ | $15.7\pm^{-}_{0.5}$ |
| | LSTM | max | LSTM | AE | $20.8\pm^{1.3}_{2.3}$ | $31.3\pm^{0.7}_{1.4}$ | $35.6\pm^{0.3}_{1.2}$ | $36.3\pm^{0.1}_{1.1}$ | $36.6\pm^{-}_{0.7}$ |
| | BoW | max | LSTM | - | $23.4\pm^{2.1}_{2.9}$ | $36.7\pm^{1.1}_{0.5}$ | $41.1\pm^{0.2}_{0.8}$ | $41.6\pm^{0.1}_{0.9}$ | $42.6\pm^{-}_{0.2}$ |
| | LSTM | max | Uni | - | $24.9\pm^{1.3}_{2.2}$ | $33.2\pm^{0.7}_{3.6}$ | $37.3\pm^{0.1}_{3.1}$ | $37.9\pm^{0.1}_{3.1}$ | $38.9\pm^{-}_{1.7}$ |
| | LSTM | last | Uni | - | $24.5\pm^{3.8}_{1.7}$ | $30.8\pm^{1.7}_{0.6}$ | $34.4\pm^{0.3}_{5.0}$ | $35.1\pm^{0.1}_{4.7}$ | $37.1\pm^{-}_{2.3}$ |
| | BoW | max | Uni | - | $24.1\pm^{2.9}_{2.7}$ | $35.0\pm^{0.9}_{1.2}$ | $39.1\pm^{0.1}_{1.8}$ | $39.5\pm^{0.1}_{1.7}$ | $40.1\pm^{-}_{0.7}$ |
| | LSTM | avg | LSTM | LM | $21.9\pm^{2.3}_{1.3}$ | $36.1\pm^{0.8}_{0.7}$ | $39.9\pm^{0.2}_{0.6}$ | $40.4\pm^{0.1}_{0.4}$ | $41.7\pm^{-}_{0.3}$ |
| Yelp | LSTM | last | LSTM | - | $49.9\pm^{4.5}_{2.7}$ | $55.6\pm^{2.3}_{2.9}$ | $57.9\pm^{1.1}_{2.5}$ | $59.5\pm^{0.2}_{2.7}$ | $61.9\pm^{-}_{2.5}$ |
| | LSTM | last | LSTM | AE | $59.3\pm^{5.4}_{2.9}$ | $80.0\pm^{1.3}_{3.0}$ | $82.7\pm^{1.0}_{0.9}$ | $83.3\pm^{0.1}_{2.3}$ | $67.9\pm^{-}_{0.1}$ |
| | LSTM | max | LSTM | - | $61.6\pm^{8.2}_{8.8}$ | $71.4\pm^{2.3}_{6.3}$ | $76.0\pm^{0.2}_{2.3}$ | $76.5\pm^{0.1}_{2.0}$ | $78.0\pm^{-}_{1.7}$ |
| | LSTM | max | LSTM | AE | $59.9\pm^{10.4}_{7.9}$ | $78.7\pm^{2.4}_{1.5}$ | $82.9\pm^{0.3}_{2.7}$ | $83.3\pm^{0.1}_{2.7}$ | $84.1\pm^{-}_{0.7}$ |
| | BoW | max | LSTM | - | $67.1\pm^{10.1}_{15.7}$ | $79.3\pm^{2.8}_{4.5}$ | $83.4\pm^{0.3}_{0.9}$ | $83.9\pm^{0.1}_{0.9}$ | $85.0\pm^{-}_{0.2}$ |
| | LSTM | max | Uni | - | $62.3\pm^{4.6}_{3.8}$ | $76.7\pm^{1.7}_{3.6}$ | $80.4\pm^{0.2}_{3.2}$ | $80.9\pm^{0.1}_{3.1}$ | $83.1\pm^{-}_{0.5}$ |
| | LSTM | last | Uni | - | $65.0\pm^{8.0}_{4.4}$ | $74.1\pm^{2.0}_{1.4}$ | $78.5\pm^{0.3}_{3.0}$ | $79.1\pm^{0.1}_{3.1}$ | $81.6\pm^{-}_{0.5}$ |
| | BoW | max | Uni | - | $59.9\pm^{7.2}_{3.7}$ | $77.3\pm^{1.2}_{0.9}$ | $81.1\pm^{0.3}_{0.5}$ | $81.5\pm^{0.1}_{0.5}$ | $83.3\pm^{-}_{0.4}$ |
| | LSTM | avg | LSTM | LM | $63.6\pm^{7.4}_{5.4}$ | $81.0\pm^{1.6}_{2.3}$ | $83.2\pm^{0.7}_{0.8}$ | $83.8\pm^{0.1}_{0.8}$ | $84.4\pm^{-}_{0.5}$ |

Table 4: Using *BoW* encoders, *Uni* decoders or *PreLM* pretraining, the representations learned by the VAEs are more predictive of the labels (sentiment or topic) of the documents.

labels correctly. We train bag-of-words classifiers[7] using either i) only the first three words or ii) all the words as features on the entire datasets. If the *three-words* classifiers are as good as the *all-words* classifiers, we expect that the original VAE variants will perform well: in that case, encoding information about the first words is not harmful, it could be a rather useful inductive bias. Conversely, if the first three words are not predictive of the label, the original VAEs will perform badly.

As reported in Table 5, on AGNews and Yelp, classifiers trained on the first 3 words have a performance somewhat close to the classifier trained on all the words, reaching 80.8% and 85.4% of its scores respectively. On AGNews, for instance, the first words are often nouns that directly gives the topic of the news item: country names for the politics category, firm names for the technology category, athlete or team names for the sports category, etc. On the two other datasets, the performance decays a lot if we only use the first three

| Dataset | F1(All) | F1(3) | Ratio |
|---|---|---|---|
| AGNews | 89.0 | 71.9 | 0.808 |
| Amazon | 48.9 | 29.7 | 0.607 |
| Yahoo | 63.0 | 19.1 | 0.303 |
| Yelp | 96.5 | 82.4 | 0.854 |

Table 5: Performance of bag-of-word classifiers when using all words as features versus only the first three words. Ratios of performance vary a lot across datasets.

[7] fastText classifiers (Joulin et al., 2017) with embedding dimension of 200 and the default parameters.

words: *three-words* F1-scores make up for 60.7% and 30.3% of *all-words* F1-scores on Amazon and Yahoo. This explains why the original VAE can perform on par or slightly better than our variants on certain datasets for which the first words are very predictive of the labels.

Despite similar asymptotic performance, the proposed variants are better than the baselines in the small data-regime, which suggests that the encoded information is quantitatively different. We will come back to this in the next evaluation.

#### 5.4.2 Recurrent and *BoW* encoders work around max-pooling

Let us focus on *BoW* encoders. It is counter-intuitive that *BoW-max-LSTM* improves over *LSTM-max-LSTM* (with or without *PreAE*). Indeed, taking into account word order should allow the LSTM encoder to do better inference than the *BoW* encoder, for example, by handling negation or parsing more complicated discourse structure (Pang et al., 2002).

We found that LSTM encoders learn an undesirable behavior through counting mechanisms (Shi et al., 2016; Suzgun et al., 2019). Indeed, they produce hidden states such that some components of the first hidden state $h_1$ *consistently* take higher values than those of $h_2, h_3, \dots$ regardless of the inputs. Similarly, other components (but in lesser quantities) are consistently maximized in the second position or the third position. Therefore, after max-pooling over these states, some components of $r$ act like memory slots assigned to fixed positions in the sentence independently of the inputs. Since the decoder is also an LSTM and can count, it extracts the relevant components at each position to retrieve the corresponding words.

Unfortunately, *BoW* encoders are not immune to this problem either. Depending on the language of the texts, the dataset and its preprocessing, the vocabulary can sometimes be partitioned in 1) words that appear in first positions of sentences and 2) the other words. For example, in English, only uppercased words will appear in the first position. Word embeddings of words that frequently start sentences can therefore learn to be identifiable by having high values at certain, fixed components, so that it is possible to identify the first word from the max-pooled representation $r$.

Therefore, it seems that the decoder and the loss play a larger role in what the encoder will learn than the encoder itself. This is rather intuitive given that the gradients of the parameters of the encoders are a function of the gradients of the decoder. This also confirms the findings of McCoy et al. (2019), who analyzed representations learned by sequence-to-sequence models (without any constraints on their capacity, see Appendix D.2).

As LSTM encoders can count, they can also easily encode sentence length. However, what happens when the sentence representation $r$ is obtained via max or average pooling on word embeddings? Assuming that a given component $j$ of the word embeddings $e_1^j$, ..., $e_L^j$ are independently distributed, then $r^j = \max_i e_i^j$ is positively correlated with $L$. If instead we assume that the $e_i^j$ have 0 mean and that $r^j = \frac{1}{L} e_i^j$, then $|r^j|$ is anti-correlated with $L$. Therefore, if the decoder encourages sentence length to be encoded, the encoder manages to do so (at least approximately) even in the absence of an explicit counting mechanism.

In summary, these experiments show that our variants encode more global information pertaining to sentiment or topic than the baselines. We have explained how the counting mechanism of the LSTM underlies memorization and how *BoW* encoders coupled with LSTM decoders are also affected by the problem, demonstrating the importance of the decoder among other architectural choices. Methodologically, we showed that only samples $z$ can be used to evaluate representations for the purpose of generation. Moreover, we stressed the importance of using different datasets, because when global attributes are not very correlated with the first words, the original VAE suffers more from its bad inductive bias.

## 6 Text generation evaluation

What is the influence of the different representations learned by our models on generation? The samples $z$ are predictive of the labels so they should also be predictive of the words that indicate the labels. Therefore, we expect that the better the classification performance, the more the reconstructed texts should exhibit the characteristics of texts sharing the same label.

To measure the agreement in label between the source document and its reconstruction, we adapt the evaluation procedure used by Ficler and Goldberg (2017) so that no human annotators or heuristics are required (see Appendix D.2). First, a classifier is trained to predict the label on the source dataset. Then, for each model, we encode the doc-

| | Enc. | $r$ | Pre. | Agreement | 1st (%) | Len (%) | NLL |
|---|---|---|---|---|---|---|---|
| AGNews | LSTM | last | AE | $80.2 \pm 1.0$ | $29.6 \pm 1.1$ | $3.6 \pm 0.1$ | $\mathbf{128.3 \pm 0.4}$ |
| | LSTM | max | AE | $79.5 \pm 0.9$ | $31.7 \pm 1.1$ | $3.7 \pm 0.5$ | $\mathbf{128.2 \pm 0.4}$ |
| | BoW | max | - | $78.0 \pm 1.3$ | $18.9 \pm 1.2$ | $\mathbf{2.7 \pm 0.3}$ | $129.5 \pm 0.6$ |
| | BoW | max | Uni | $81.3 \pm 0.1$ | $\mathbf{13.9 \pm 0.3}$ | $3.1 \pm 0.1$ | $129.7 \pm 0.7$ |
| | LSTM | max | Uni | $\mathbf{82.0 \pm 0.4}$ | $\mathbf{13.9 \pm 0.2}$ | $3.3 \pm 0.4$ | $129.4 \pm 0.4$ |
| | LSTM | avg | LM | $79.2 \pm 0.4$ | $22.2 \pm 0.8$ | $3.2 \pm 0.2$ | $\mathbf{128.4 \pm 0.3}$ |
| Amazon | LSTM | last | AE | $24.5 \pm 0.4$ | $42.4 \pm 2.3$ | $13.0 \pm 1.6$ | $\mathbf{82.8 \pm 0.1}$ |
| | LSTM | max | AE | $30.8 \pm 1.1$ | $41.7 \pm 0.8$ | $11.5 \pm 1.0$ | $\mathbf{82.8 \pm 0.1}$ |
| | BoW | max | - | $34.2 \pm 0.5$ | $33.3 \pm 0.7$ | $\mathbf{9.9 \pm 0.7}$ | $83.2 \pm 0.2$ |
| | BoW | max | Uni | $33.3 \pm 0.4$ | $\mathbf{21.5 \pm 0.3}$ | $11.8 \pm 0.5$ | $83.2 \pm 0.2$ |
| | LSTM | max | Uni | $34.1 \pm 0.5$ | $22.1 \pm 0.1$ | $11.7 \pm 0.6$ | $83.3 \pm 0.3$ |
| | LSTM | avg | LM | $\mathbf{35.8 \pm 0.4}$ | $38.3 \pm 0.9$ | $11.5 \pm 1.0$ | $\mathbf{82.7 \pm 0.2}$ |
| Yahoo | LSTM | last | AE | $23.8 \pm 0.2$ | $56.6 \pm 1.0$ | $17.1 \pm 1.1$ | $\mathbf{81.4 \pm 0.1}$ |
| | LSTM | max | AE | $22.9 \pm 0.8$ | $58.7 \pm 1.7$ | $18.4 \pm 0.8$ | $\mathbf{81.3 \pm 0.1}$ |
| | BoW | max | - | $\mathbf{26.9 \pm 0.5}$ | $49.3 \pm 1.2$ | $11.8 \pm 0.3$ | $81.8 \pm 0.2$ |
| | BoW | max | Uni | $\mathbf{26.8 \pm 0.6}$ | $\mathbf{37.6 \pm 0.9}$ | $\mathbf{10.6 \pm 0.4}$ | $81.8 \pm 0.0$ |
| | LSTM | max | Uni | $\mathbf{27.1 \pm 1.0}$ | $\mathbf{37.7 \pm 1.6}$ | $\mathbf{11.0 \pm 0.3}$ | $81.9 \pm 0.2$ |
| | LSTM | avg | LM | $\mathbf{26.7 \pm 0.2}$ | $51.9 \pm 0.5$ | $16.7 \pm 1.8$ | $\mathbf{81.3 \pm 0.1}$ |
| Yelp | LSTM | last | AE | $\mathbf{81.7 \pm 1.3}$ | $53.0 \pm 0.5$ | $33.7 \pm 1.7$ | $\mathbf{34.3 \pm 0.1}$ |
| | LSTM | max | AE | $81.3 \pm 0.7$ | $52.4 \pm 0.5$ | $29.5 \pm 2.5$ | $\mathbf{34.4 \pm 0.0}$ |
| | BoW | max | - | $\mathbf{82.2 \pm 0.5}$ | $36.4 \pm 0.3$ | $22.4 \pm 0.5$ | $34.5 \pm 0.1$ |
| | BoW | max | Uni | $80.4 \pm 0.4$ | $\mathbf{30.6 \pm 0.5}$ | $\mathbf{15.4 \pm 0.4}$ | $34.7 \pm 0.0$ |
| | LSTM | max | Uni | $80.9 \pm 0.4$ | $32.0 \pm 0.4$ | $17.2 \pm 0.7$ | $34.8 \pm 0.1$ |
| | LSTM | avg | LM | $\mathbf{82.3 \pm 0.7}$ | $47.7 \pm 0.4$ | $24.1 \pm 0.4$ | $\mathbf{34.4 \pm 0.1}$ |

Table 6: Our variants reconstruct the inputs with 1) higher agreement with the ground-truth, 2) less memorization of the 1st word and the length, 3) with a negligible loss in likelihood. The best score and scores within one std are bolded.

uments, reconstruct them, and classify these reconstructions using the classifier. Finally, we report the F1 scores between the original labels and the labels given by the classifiers on the generated samples. We call this score the *agreement*.

We use two decoding schemes: beam search with a beam of size 5 and greedy decoding. We fix $\lambda = 8$, $d = 16$ on all models with three seeds. For the *Uni* decoder, we drop *LSTM-last-Uni* which underperformed by a large margin in the SSL setting, and for the other *Uni* models, we freeze the encoder, $L_1$ and $L_2$ and train a new recurrent decoder using the reconstruction loss of the VAE. The *Uni* decoder is used as an auxiliary decoder, as described by De Fauw et al. (2019) (see Appendix D.1 for details) and we denote this technique by *PreUni*.

To quantify memorization, we measure the reconstruction accuracy of the first word and the ratio of identical sentence length between sources and reconstructions, as in Table 2. Finally, to verify that our bag-of-words assumptions do not hurt the overall fit to the data, we estimate the negative log-likelihood (*NLL*) via the importance-weighted lower bound (Burda et al., 2015) (500 samples).

Table 6 show the results for beam search decoding.[8] There is a close correspondence between agreement and performance on the SSL tasks in the large data-regime. Our variants have higher agreement than the baseline, especially on Amazon and Yahoo datasets where, as we have seen before, the memorization of the first words is an especially bad inductive bias. Note that on these datasets, the agreements are consistently lower than the downstream task performance classification, which shows that reconstructing a sentence with the same label as the source sentence is harder than predicting the label using a classifier. Apart from that, the agreement does not tell us much more than the SSL results.

However, the baselines reconstruct the first words with very high accuracy (more than 50% of the time on Yahoo and Yelp) while our variants mitigate this memorization. For instance, the *PreUni* method recovers the first word around twice less often on AGNews and Amazon and 1.5 less often on Yahoo and Yelp. This is particularly interesting on AGNews and Yelp, where the first words

[8] Similar results were obtained using greedy decoding, albeit sometimes consistently shifted.

are very indicative of the topics or sentiments, both baselines and variants have similarly high agreement. This shows that the mechanisms to produce texts with the same labels are different: the reconstructions of the baselines exhibit the same labels as the sources mostly as a side-effect of starting with the same words. On the other hand, our best variants have more diverse beginning of sentences but nonetheless produce as many or more documents of the correct labels.

We can now interpret the discrepancy of results between the small and large data-regimes that we have observed in the SSL setting. Recall that despite similar performances using a lot of data, our variants were much more efficient using very few labels (5 examples per class). If the baselines simply memorize the first words of the sentences by mapping prefixes of the sentences (possibly of varying sizes) to latent vectors, the amount of data required to learn a good classifier will be higher than if the features are more global and abstract.

Swapping the LSTM encoder with a *BoW* encoder yields less memorization of the first word; further swapping the LSTM decoder with a *Uni* decoder decreases memorization further. This shows that our bag-of-words assumptions, both on the encoder and the decoder side, are efficient for dealing with the memorization problem. Note that *BoW-Max* and *LSTM-Max* with *PreUni* pretraining yield very close performance despite having a different encoder, which confirms that the choice of the decoder is much more important than the choice of the encoder.

Finally, there seems to be a tradeoff between the global character of the latent information and the fit of the model to the data, as *BoW* and *Uni* variants have a higher negative log-likelihood than the baselines. The difference seems significant (informally speaking, by looking at the standard deviations) but the effect size is very small and should not impact the overall quality of the generated texts.

To recapitulate, the bag-of-words assumptions decrease the memorization of the first word and of the sentence length in the latent variable while increasing the agreement between the labels of the source and of the reconstruction. This is achieved at the cost of a very small decrease in log-likelihood.

## 7 Conclusion and outlook

Since the inception of the sequence-to-sequence VAE, a lot of efforts were invested in solving the posterior collapse problem and learning to encode *something*. However, this is not a sufficient condition for VAEs to be used for SSL or controllable generation, use cases for which latent variables should encode global information. By decomposing the reconstruction loss per positions in the documents, we showed that sequence-to-sequence VAEs, both the original versions and recent variants, tend to memorize the first few words as well as the length of the documents. These VAEs sometimes capture global features, but coincidentally and as a side-effect of their memorization behavior, when these features are correlated with the first words of the documents.

In order to reduce memorization, we proposed simple modifications to the architecture (bag-of-words encoders or unigram decoders) and to the training procedure (pretraining with a language modelling objective). In the semi-supervised learning setting, our simple variants produce representations that are more predictive of the ground-truth labels and these gains translate directly in generation. We obtained a higher agreement between the labels of source texts and the labels of their reconstructions with less memorization at almost no cost in terms of likelihood.

A lot of work remains to be done. The root cause of memorization should be clearly identified. A first hypothesis to explore is that the fixed, left-to-right factorization of the probability of the decoder could lead to memorization of the first words. Indeed, on all datasets, the *LSTM-LM* incurs a higher reconstruction loss on the first positions (cf. Figure 1) and these early errors should account for a proportionally larger part of the gradients. This hypothesis is also supported by our successes with the unigram decoder, which models words independently. If the hypothesis were true, we would expect that either non-autoregressive decoders (for instance Gu et al., 2017) or auto-regressive models where the order is latent and therefore, variable (for example, Gu et al., 2019) would not exhibit memorization. It would also imply that standard Transformers used auto-regressively would not yield improvements. Similarly, the causes behind the encoding of sentence length should be analyzed in depth.

Another promising avenue is to draw inspiration from models, training procedures and losses used for language model pretraining. Models such as BERT (Devlin et al., 2018) *only* penalize the re-

construction of the words that are either missing or corrupted and therefore, they avoid memorization altogether. These models can be seen as denoising auto-encoders (*DAE*) (Vincent et al., 2008). Current VAE models learn to corrupt the latent space and to reconstruct the entire input, while current DAE models corrupts parts of their inputs and reconstruct the corrupted portions of their inputs. Models which blend the two frameworks might have the best of both worlds (Im et al., 2017).

## References


Alexander Alemi, Ben Poole, Ian Fischer, Joshua Dillon, Rif A Saurous, and Kevin Murphy. 2018. Fixing a Broken ELBO. In *International Conference on Machine Learning*, pages 159–168.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics.

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.

Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. 2015. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Xi Chen, Diederik P Kingma, Tim Salimans, Yan Duan, Prafulla Dhariwal, John Schulman, Ilya Sutskever, and Pieter Abbeel. 2016. Variational lossy autoencoder. *arXiv preprint arXiv:1611.02731*.

Jeffrey De Fauw, Sander Dieleman, and Karen Simonyan. 2019. Hierarchical autoregressive image models with auxiliary decoders. *arXiv preprint arXiv:1903.04933*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Jessica Ficler and Yoav Goldberg. 2017. Controlling Linguistic Style Aspects in Neural Language Generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Hao Fu, Chunyuan Li, Xiaodong Liu, Jianfeng Gao, Asli Celikyilmaz, and Lawrence Carin. 2019. Cyclical Annealing Schedule: A Simple Approach to Mitigating KL Vanishing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 240–250, Minneapolis, Minnesota. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2017. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*.

Jiatao Gu, Qi Liu, and Kyunghyun Cho. 2019. Insertion-based decoding with automatically inferred generation order. *Transactions of the Association for Computational Linguistics*, 7:661–676.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging Inference Networks and Posterior Collapse in Variational Autoencoders. In *International Conference on Learning Representations*.

Sepp Hochreiter and Jrgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780.

Daniel Im Jiwoong Im, Sungjin Ahn, Roland Memisevic, and Yoshua Bengio. 2017. Denoising criterion for variational auto-encoding framework. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. Bag of Tricks for Efficient Text Classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain. Association for Computational Linguistics.

Diederik P. Kingma, Danilo Jimenez Rezende, Shakir Mohamed, and Max Welling. 2014. Semi-Supervised Learning with Deep Generative Models. *CoRR*, abs/1406.5298.

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

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

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A Surprisingly Effective Fix for Deep Latent Variable Modeling of Text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3601–3612, Hong Kong, China. Association for Computational Linguistics.

Teng Long, Yanshuai Cao, and Jackie Chi Kit Cheung. 2019. Preventing Posterior Collapse in Sequence VAEs with Pooling. *arXiv:1911.03976 [cs, stat]*. ArXiv: 1911.03976.

R. Thomas McCoy, Tal Linzen, Ewan Dunbar, and Paul Smolensky. 2019. RNNs implicitly implement tensor-product representations. In *International Conference on Learning Representations*.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural Variational Inference for Text Processing. In *International Conference on Machine Learning*, pages 1727–1736.

Henry B Moss, David S Leslie, and Paul Rayson. 2018. Using JK fold Cross Validation to Reduce Variance When Tuning NLP Models. *arXiv preprint arXiv:1806.07139*.

Gary W Oehlert. 2010. *A first course in design and analysis of experiments*.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

Tom Pelsmaeker and Wilker Aziz. 2019. Effective Estimation of Deep Generative Language Models. *arXiv preprint arXiv:1904.08194*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Victor Prokhorov, Ehsan Shareghi, Yingzhen Li, Mohammad Taher Pilehvar, and Nigel Collier. 2019. On the Importance of the Kullback-Leibler Divergence Term in Variational Autoencoders for Text Generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 118–127, Hong Kong. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. Technical report.

Ali Razavi, Aaron van den Oord, Ben Poole, and Oriol Vinyals. 2019. Preventing Posterior Collapse with delta-VAEs. In *International Conference on Learning Representations*.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic Backpropagation and Approximate Inference in Deep Generative Models. In *International Conference on Machine Learning*, pages 1278–1286.

Xing Shi, Kevin Knight, and Deniz Yuret. 2016. Why Neural Translations are the Right Length. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2278–2282, Austin, Texas. Association for Computational Linguistics.

Paul Smolensky. 1990. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial intelligence*, 46(1-2):159–216.

Mirac Suzgun, Yonatan Belinkov, Stuart Shieber, and Sebastian Gehrmann. 2019. LSTM Networks Can Perform Dynamic Counting. In *Proceedings of the Workshop on Deep Learning and Formal Languages: Building Bridges*, pages 44–54, Florence. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level Convolutional Networks for Text Classification. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems 28*, pages 649–657. Curran Associates, Inc.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning Discourse-level Diversity for Neural Dialog Models using Conditional Variational Autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

## A On the use of KL annealing, the choice of the free bits flavor and resetting the decoder

Li et al. (2019) evaluated their models in the SSL setting using relatively small training sets (Section 3.3 of their paper). However, their experimental setting is not very rigorous. They use a validation set containing 10000 examples to do model selection, which is also the size of their largest training set. This is equivalent to selecting the model on the test set. The hyperparameter budget seems to be different for different models, exacerbating the problem. Finally, it seems that KL annealing played the same role as the free bits technique and that therefore, KL annealing was redundant. Therefore, we run our own hyperparameter search on the Yelp dataset.

Our experiments clearly confirm that their pretraining technique improves the performance, but their choice of the free bits technique and the use of KL annealing is suboptimal. We first show that KL annealing is not necessary anymore when we use free bits, and that the original free bits method is equivalent or worse than the $\delta$-VAE variant. This justifies our use of a slightly different method than their method in the paper. Additionally, we also confirm that resetting the decoder is crucial.

### A.1 The free bits technique and variants

The *original* free bits objective (Kingma et al., 2016) is the following modification to the KL term:

$$\sum_{j}^{K} \max(\frac{\lambda}{K}, KL(q_j(z_j|x)||p_j(z_j)))$$

where indices denote components. In this formulation, each component of the multivariate normal is allowed to deviate from the prior by a small amount. Instead, in the $\delta$-VAE formulation, one component can use of all the $\lambda$ free bits and the rest of the components can collapse to the prior. This is the variant called $\delta$, used throughout the paper:

$$\max(\lambda, KL(q(z|x)||p(z)))$$

Other modifications of the free bits technique include the use of a variable coefficient in front of the KL term (Chen et al., 2016), the target rate objective in Alemi et al. (2018), minimum desired rate (Pelsmaeker and Aziz, 2019), etc. A comparison of all these methods is out of the scope of this paper and the $\delta$ variant satisfies our only requirement: the rate should be close to the desired rate.

### A.2 KL annealing and the original free bits method higher the rate

Our hypothesis is that KL annealing is redundant when used with free bits. Therefore, it should increase the actual rate more than with free bits alone. This prevents comparisons of models fairly, at equivalent capacity. We also posit that the original free bits formulation impose unnecessary constraints on how the free bits should be use, namely, that they should be used equally in all components.

To study the influence of the free bits variant as well as of KL annealing, we use the same experimental protocol as described in Section 5. To save computations, we fix $d = 16$. We do not perform model selection on the desired rate $\lambda$ in order to see which methods yield the rates that are closest to the desired rate.

Table 7 shows that both KL annealing and the original free bits term instead of the $\delta$-VAE variant increase the actual rate that is reached at the end of the optimisation. Moreover, the increases are very unpredictable: we gain higher capacity due to using KL annealing when we are using the original free bits than when we are not. Therefore, we cannot hope to do comparisons with equal rates using KL annealing. The $\delta$-VAE free bits variant without annealing reach the closest KL value to the desired target rate $\lambda$. In addition, the $\delta$-VAE free bits without KL annealing consistently yield better downstream task performance. In summary, KL annealing is harmful when used with free bits and the $\delta$-VAE free bits technique is superior to the original formulation. Therefore, all the experiments in the paper use the $\delta$ variant without annealing.

In Li et al. (2019)'s work, the "per-component" variant might have been chosen because it **trivially** maximizes a metric called *active units* (AU). This measure quantifies roughly how many components of the latent vector deviates by a certain threshold from the prior on average. However, to our knowledge, there is no evidence that this metric should be maximized, neither theoretical nor empirical. Arguably, it is not only meaningless but also detrimental to maximize this metric as it discourages sparsity. Hence, we refrain from using this metric.

## B On the importance of resetting the decoder after pretraining

Li et al. (2019) proposed to pretrain an *AE* with a reconstruction loss only. Then, the parameters of the decoder are re-initialised and the (modified) KL term is added to the objective. Since it is not very clear why it would be useful, we studied the impact of this choice. Table 8 shows that it is is crucial.

## C Training procedure

All the runs are trained using SGD with a learning rate of $0.5$ and gradients are clipped when their norms are higher than 5. We use the following early stopping scheme: at every epoch, if there has not been improvements on the validation error for two epochs in a row, the learning rate is halved. Once it has been halved four times, the training stops.

All the LSTMs have hidden state size of 512 and

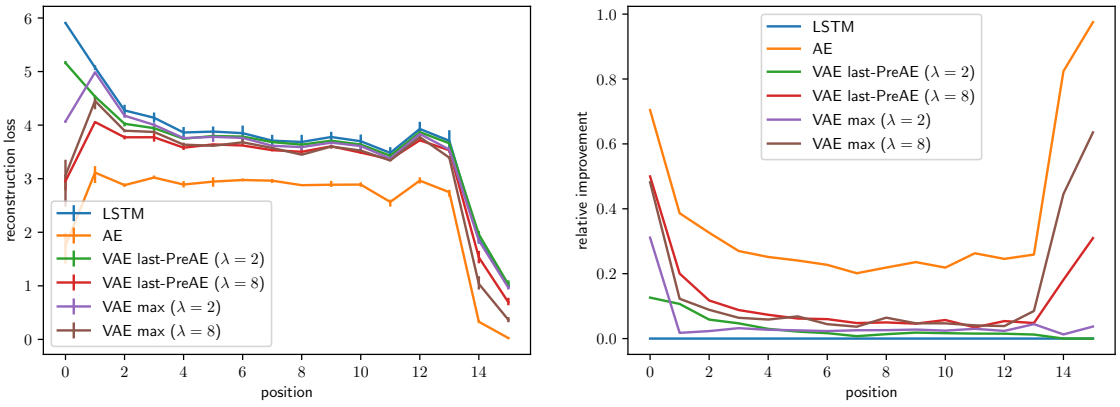


Figure 2: Reconstruction loss as a function of word position on the AGnews dataset. See Figure 1.

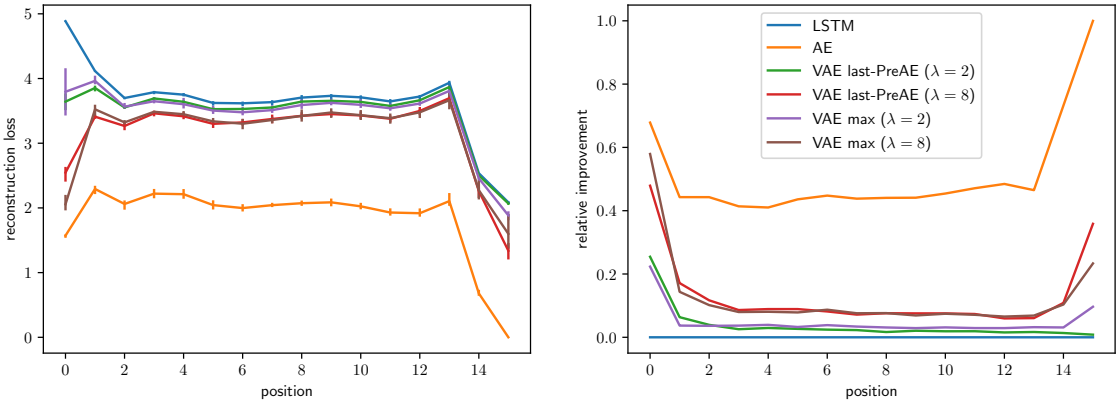


Figure 3: Reconstruction loss as a function of word position on the Amazon dataset. See Figure 1.

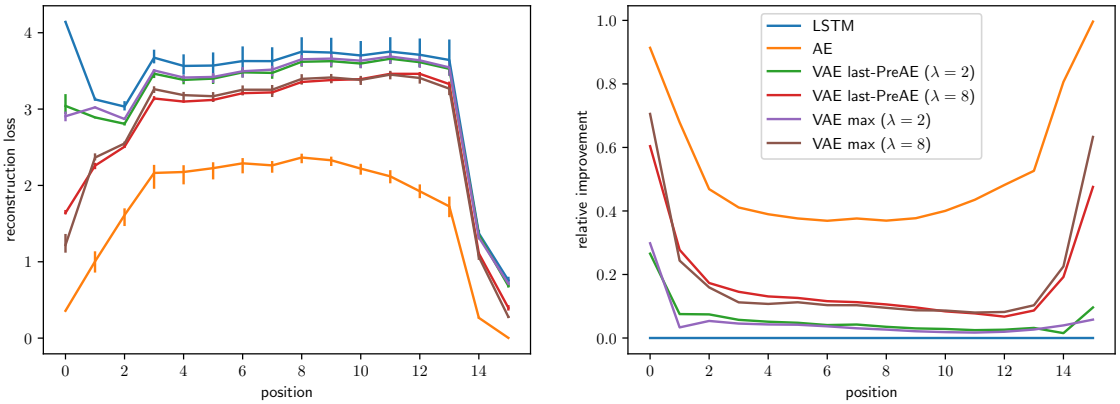


Figure 4: Reconstruction loss as a function of word position on the Yahoo dataset. See Figure 1.

| FB | $\lambda$ | ANN. | F1(5) | F1(50) | F1(500) | F1(5000) | F1(ALL) | KL(5) |
|---|---|---|---|---|---|---|---|---|
| O | 2 | 10 | $53.3\pm^{5.5}_{3.3}$ | $69.8\pm^{1.8}_{1.3}$ | $73.6\pm^{0.2}_{1.7}$ | $74.0\pm^{0.1}_{1.8}$ | $73.6\pm_{1.1}$ | $5.27\pm_{0.47}$ |
| O | 2 | 0 | $51.8\pm^{4.8}_{6.7}$ | $62.7\pm^{2.5}_{3.8}$ | $67.0\pm^{0.4}_{5.6}$ | $67.5\pm^{0.1}_{5.8}$ | $66.9\pm_{2.7}$ | $2.58\pm_{0.46}$ |
| $\delta$ | 2 | 10 | $51.7\pm^{4.6}_{4.7}$ | $64.5\pm^{1.9}_{6.7}$ | $68.3\pm^{0.4}_{7.3}$ | $69.1\pm^{0.2}_{6.7}$ | $68.4\pm_{3.3}$ | $2.5\pm_{0.24}$ |
| $\delta$ | 2 | 0 | $58.7\pm^{5.5}_{3.2}$ | $74.0\pm^{2.7}_{4.4}$ | $78.1\pm^{0.3}_{4.1}$ | $78.6\pm^{0.1}_{4.3}$ | $78.6\pm_{1.9}$ | $2.27\pm_{0.02}$ |
| O | 8 | 10 | $60.0\pm^{6.0}_{8.7}$ | $77.5\pm^{1.2}_{2.2}$ | $80.8\pm^{0.3}_{4.1}$ | $81.2\pm^{0.1}_{4.2}$ | $81.2\pm_{2.1}$ | $10.67\pm_{0.44}$ |
| O | 8 | 0 | $60.2\pm^{7.3}_{4.7}$ | $77.7\pm^{2.0}_{2.6}$ | $81.4\pm^{0.3}_{2.2}$ | $81.7\pm^{0.1}_{2.2}$ | $81.5\pm_{0.9}$ | $9.48\pm_{0.08}$ |
| $\delta$ | 8 | 10 | $57.6\pm^{7.6}_{4.2}$ | $76.3\pm^{1.4}_{1.1}$ | $80.3\pm^{0.3}_{3.0}$ | $80.8\pm^{0.1}_{2.9}$ | $80.3\pm_{1.0}$ | $8.21\pm_{0.07}$ |
| $\delta$ | 8 | 0 | $60.4\pm^{4.1}_{3.6}$ | $80.0\pm^{1.3}_{3.0}$ | $82.7\pm^{1.0}_{0.9}$ | $83.3\pm^{0.1}_{2.3}$ | $83.5\pm_{0.8}$ | $8.12\pm_{0.02}$ |

Table 7: $\delta$-VAE-style free bits with no KL annealing delivers the best downstream task performance and a KL closest to the desired rate. *Ann.*: 0: no annealing, 10: anneal for 10 epochs; *FB*: free bits type; *F1($n$)*: F1 score in the $n$ data-regime; *KL*: rate obtained after training.

| RESET. | $\lambda$ | F1(5) | F1(50) | F1(500) | F1(5000) | F1(ALL) | KL(5) |
|---|---|---|---|---|---|---|---|
| N | 2 | $51.0\pm^{4.2}_{5.6}$ | $61.3\pm^{2.0}_{9.2}$ | $65.6\pm^{0.5}_{9.2}$ | $66.2\pm^{0.1}_{9.5}$ | $65.2\pm_{4.9}$ | $2.36\pm_{0.15}$ |
| Y | 2 | $58.7\pm^{5.5}_{3.2}$ | $74.0\pm^{2.7}_{4.4}$ | $78.1\pm^{0.3}_{4.1}$ | $78.6\pm^{0.1}_{4.3}$ | $78.6\pm_{1.9}$ | $2.27\pm_{0.02}$ |
| N | 8 | $57.4\pm^{5.6}_{2.4}$ | $73.4\pm^{1.5}_{7.3}$ | $77.2\pm^{0.3}_{6.6}$ | $77.5\pm^{0.1}_{6.7}$ | $77.4\pm_{2.6}$ | $8.23\pm_{0.08}$ |
| Y | 8 | $60.4\pm^{4.1}_{3.6}$ | $80.0\pm^{1.3}_{3.0}$ | $82.7\pm^{1.0}_{0.9}$ | $83.3\pm^{0.1}_{2.3}$ | $83.5\pm_{0.8}$ | $8.12\pm_{0.02}$ |

Table 8: Resetting the decoder brings very noticeable gains on all data-regimes and with different rates. Yelp dataset, $\delta$-VAE free bits, no KL annealing. For columns interpretations, see Table 7.

use a batch size of 64. No dropout is applied to the encoders. The LSTM decoders use dropout ($p = 0.5$) both on embeddings and on the hidden states (before the linear transformation that gives logits). Similarly, dropout is applied to the representation before the linear transformation that gives the logits for the Unigram decoder. Word embeddings are initialized randomly and learned.

# D Related work

## D.1 Related models

The models that we use are very similar to already proposed models.

The NVDM model of Miao et al. (2016) is precisely *BoW-max-Uni*.

Zhao et al. (2017) proposed to use an auxiliary loss that consists in reconstructing the input using a unigram model. Thus, their objective contains two reconstruction losses: the reconstruction loss given by the recurrent decoder and the one given by the unigram decoder. In comparison, our *Uni* models are trained in two steps: the encoder is trained jointly with the unigram decoder, then the decoder is thrown away and we train a recurrent decoder using the fixed encoder. This way, we do not fear that one decoder might dominate the other and moreover, we do not deal with potential hyperparameters that weigh the two losses. Instead of having an auxiliary *loss*, we have an auxiliary *decoder* that is only used for the purpose of training the encoder. This method was presented by De Fauw et al. (2019) for training generative models of image. There is a slight difference: they use a feedforward auxiliary decoder to produce *different* probability distributions for all the pixels, whereas our unigram probability distribution is *the same* for all words of a document. This modification allows us to deal with varying lengths of documents.

Finally, the *PreLM* training procedure is related to large LM pretraining in the spirit of contextualized embeddings (Peters et al., 2018) and its successors. Note, however, two differences. Firstly, we do not use external data and stick to each individual training set. The goal is obviously not to evaluate transfer learning abilities. Secondly, we do not fine-tune the entire encoder but merely learn the linear transformations $L_1$ and $L_2$ that produce the variational parameters, to make sure that the VAE objective will have no impact on the extraction of features.

## D.2 Methods and evaluations

Ficler and Goldberg (2017) learn *LSTM-LMs* conditioned on labels that describe high-level properties of texts. Among others, they want to verify that generated texts exhibit the same properties as the conditioning labels. For instance, when the *LSTM-LM* is conditioned on positive sentiment value, the

generated texts should also exhibit a positive sentiment. To check that the conditioning variables and the generated texts are consistent, they use the following procedure. First, they extract information about the various documents using heuristics or with the help of annotators. Then, they learn *LSTM-LM*s conditioned on these labels. Finally, they quantify the ratio of generated samples which have the same labels than the conditioning labels, either by applying the same heuristics again to the generated samples or by asking human annotators once more. Our evaluation in Section 6 is extremely similar. We simply replace the heuristics and the human annotators with classifiers learned on ground-truth data.

Our work is also related to the important work of McCoy et al. (2019). They trained auto-encoders with different combinations of encoders and decoders (unidirectional, bidirectional or tree-structured) and decomposed the representations learned by the encoders using tensor product representations (Smolensky, 1990). They showed that decoders "largely dictate" the way information is encoded. The main difference between our works is that they study how information is encoded in sequence-to-sequence models without capacity limitations, whereas we study what information is encoded in the VAE sequence-to-sequence model, where the VAE objective puts severe limits on capacity.

## E Decomposing the variances of the scores

For a given model, dataset and data-regime, after optimisation of the hyperparameters of the VAE and the classifier, we collect several F1-scores $F_{ij}$ which depend on the seed used to subsample the dataset $i$ and the seed used to initialise the model parameters $j$. We posit a linear model with one random-effect factor, the initialisation seed, and where replicates are obtained by varying the subsampling seed:

$$F_{ij} = \mu + \alpha_j + \epsilon_{ij}$$

Assuming that $\alpha_j$ and $\epsilon_{ij}$ are independent random variables with null expectations, we can decompose the variance as

$$\begin{aligned}\mathrm{Var}(F_{ij}) &= \mathbb{E}[(F_{ij} - \mu)^2] \\ &= \mathbb{E}[(\alpha_i + \epsilon_{ij})^2] \\ &= \mathbb{E}[\alpha_i^2] + \mathbb{E}[\epsilon_{ij}^2] \\ &= \mathrm{Var}(\alpha_i) + \mathrm{Var}(\epsilon_{ij})\end{aligned}$$

This is the basis of the method of analysis of variance (ANOVA) and is often used to test hypotheses (for instance, that the effect $\mathbb{E}[\alpha_i]$ is significant) (Oehlert, 2010). The two estimates of $\sigma^2_{init}$ and $\sigma^2$ are usually denoted $MS_T$ and $MS_E$.

In our case, we are only interested in estimating roughly what variability is due to the model initialisation and what is due to the subsampling of the dataset.

Note that we could treat the two sources of variance $i$ and $j$ symmetrically by adding add a term $\beta_i$, but we would need to report 3 standard deviations (that of $\alpha_j$, $\beta_i$ and $\epsilon_{ij}$) to get the full picture. The most important estimate is $\sigma_{init}$. It quantifies the inherent robustness of the model to different initialisations. The effect of the subsampling is specific to the dataset, therefore, it is less relevant to our analysis.