Learning Sparse Sentence Encoding without Supervision: An Exploration of Sparsity in Variational Autoencoders

Victor Prokhorov[♣] Yingzhen Li[◇*] Ehsan Shareghi[♠♣] Nigel Collier[♣]

[♣] Language Technology Lab, University of Cambridge

[♠] Department of Data Science & AI, Monash University

[◇] Department of Computing, Imperial College London

vp361@cam.ac.uk, yingzhen.li@imperial.ac.uk, ehsan.shareghi@monash.edu, nhc30@cam.ac.uk

Abstract

It has been long known that sparsity is an effective inductive bias for learning efficient representation of data in vectors with *fixed dimensionality*, and it has been explored in many areas of representation learning. Of particular interest to this work is the investigation of the sparsity within the VAE framework which has been explored a lot in the image domain, but has been lacking even a basic level of exploration in NLP. Additionally, NLP is also lagging behind in terms of learning sparse representations of large units of text e.g., sentences. We use the VAEs that induce sparse latent representations of large units of text to address the aforementioned shortcomings. First, we move in this direction by measuring the success of unsupervised state-of-the-art (SOTA) and other strong VAE-based sparsification baselines for text and propose a hierarchical sparse VAE model to address the stability issue of SOTA. Then, we look at the implications of sparsity on text classification across 3 datasets, and highlight a link between performance of sparse latent representations on downstream tasks and its ability to encode task-related information.[1]

1 Introduction

Representation learning has been pivotal in many success stories of modern days NLP. Observing its success, two fundamental questions arise: *How is the information encoded in them?* and *What is encoded in them?* While the latter has received a lot of attention by designing probing tasks, the former has been vastly neglected. In this work, we take small steps in this non-trivial direction by building on the knowns: One property we know about the encoding of information is that different data points embody different characteristics (e.g. statistically, semantically, or syntactically) which should ideally utilise different sub-regions of the representation space. Therefore, the high-dimensional learned representations should ideally be sparse (Bengio et al., 2013; Burgess et al., 2018; Tonolini et al., 2019). In other words it allows us to have varying number of active dimension per sentence[2] (Bengio, 2009) in a fixed dimensional vector[3]. But *if sparsity*[4] *is expected, could it be learned from data without supervision?*

A handful of studies in NLP that have delved into building sparse representations of words either during the learning phase (Faruqui and Dyer, 2015; Yogatama et al., 2015) or as a post-processing step on top of existing representations (e.g., word2vec embeddings) (Faruqui et al., 2015; Sun et al., 2016; Subramanian et al., 2018; Arora et al., 2018; Li and Hao, 2019). These methods have not been developed for sentence embeddings, with the exception of Trifonov et al. (2018) which makes a strong assumption by forcing the latent sentence representation to be a sparse categorical distribution.

In parallel, Variational Autoencoders (VAEs) (Kingma and Welling, 2014) have been effective in capturing semantic closeness of sentences in the learned representation space (Bowman et al., 2016; Prokhorov et al., 2019; Xu et al., 2019; Balasubramanian et al., 2020). Furthermore, methods have been developed

[*] Work done while at Microsoft Research Cambridge.

[1] The code is available on https://github.com/VictorProkhorov/HSVAE.

[2] This, for example, may allow us to cluster sentences' representations not only based on similarity of their active features (as it is the case for dense vectors) but also on active/inactive dimensions.

[3] More on speculative side, sparse representations may be a more natural way of modelling sentences of a language in a fixed dimensional vector. Sentences vary in length and an amount of information that they convey. As such it makes sense to reflect this property in a vector representation of the sentence.

[4] As in (Mathieu et al., 2019), we induce sparse representations for each data point.

to encourage sparsity in VAEs via learning a deterministic selection variable (Yeung et al., 2017) or sparse priors (Barello et al., 2018; Mathieu et al., 2019; Tonolini et al., 2019). However, the success of these is yet to be examined on text domain.

To bridge this gap, we make a sober evaluation of existing state-of-the-art (SOTA) VAE-based sparsification model (Mathieu et al., 2019) against several VAE-based baselines on two experimental tasks: text classification accuracy, and the level of representation sparsity achieved. Additionally, we propose Hierarchical Sparse Variation Autoencoder (HSVAE), to improve the stability issue of existing SOTA model and demonstrate its performance on both experimental tasks.

Our experimental findings demonstrate that: (I) neither the simpler baseline models nor the SOTA manage to impose a satisfactory level of sparsity on text, (II) as expected, sparsity level and task performance have a negative correlation, while giving up task performance and having sparse codes helps with the analysis of the representations, (III) presence/absence of task related signal in the sparsity codes affects the task performance, (IV) the success of capturing the task related signal in the sparsity codes depends on the strength of the signal presented in a corpus, and representation dimensionality, (V) the success of SOTA in image domain does not necessarily transfer to inducing sparse representations for text, while HSVAE addresses this shortcoming.

2 Background

VAE. Given an input x, VAEs, Figure 1 (left), are stochastic autoencoders that map x to a corresponding representation z using a probabilistic encoder $q_\phi(z|x)$ and a probabilistic decoder $p_\theta(x|z)$, implemented as neural networks. Optimisation of VAE is done by maximising the ELBO:

$$\mathbb{E}_{q_\phi(z|x)} \log p_\theta(x|z) - \mathbb{D}_{KL}\big(q_\phi(z|x)||p_\theta(z)\big) \quad (1)$$

where the reconstruction maximises the expectation of data likelihood under the posterior distribution of z, and the Kullback-Leibler (KL) divergence acts as a regulariser and minimises the distance between the learned posterior and prior of z.

Spike-and-Slab Distribution. This is a mixture of two Gaussians with mixture weight γ_i, where the *slab* component is a standard Gaussian while the *spike* component is a Gaussian with $\sigma \to 0$:

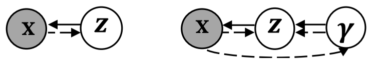


Figure 1: Graphical Models of VAE (left) and HSVAE (right). Solid and dashed lines represent generative and inference paths, respectively.

$$p(z) = \prod_i^{\mathrm{D}} (1-\gamma_i)\,\mathcal{N}(z_i; 0, 1) + \gamma_i\,\mathcal{N}(z_i; 0, \sigma \to 0)$$

where i denotes the ith dimension of z and D is the total number of dimensions of z.

3 Hierarchical Sparse VAE (HSVAE)

We propose the hierarchical sparse VAE (HSVAE), Figure 1 (right), to learn sparse latent codes automatically. We treat the mixture weights $\gamma = (\gamma_1, ..., \gamma_D)$ as a random variable and assign a factorised Beta prior $p_\theta(\gamma_i) = \mathrm{Beta}(\alpha, \beta)$ on it. The latent code z is then sampled from a factorised Spike-and-Slab distribution $p_\theta(z|\gamma)$ conditioned on γ, and the observation x is generated by decoding the latent variable $x \sim p_\theta(x|z)$ using a GRU (Cho et al., 2014) decoder. This returns a probabilistic generative model $p_\theta(x, z, \gamma) = p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)$.

For posterior inference, the encoder distribution is defined as $q_\phi(z, \gamma|x) = q_\phi(\gamma|x)q_\phi(z|\gamma, x)$, where $q_\phi(\gamma|x)$ is a learnable and factorised Beta distribution, and $q_\phi(z|\gamma, x)$ is a factorised Spike-and-Slab distribution with mixture weights γ_i and learnable "slab" components for each dimension. The q distribution is computed by first extracting features from the sequence using a GRU, then applying MLPs to the extracted feature (and γ for $q_\phi(z|\gamma, x)$) to produce the distributional parameters.

ELBO: We derive the ELBO, $\mathcal{L}(\theta, \phi; x)$:

$$\mathbb{E}_{q_\phi(z,\gamma|x)}[\log p_\theta(x|z)] - \psi\mathbb{E}_{q_\phi(\gamma|x)}[\mathbb{D}_{KL}\big(q_\phi(z|\gamma, x), p_\theta(z|\gamma)\big)] - \lambda\mathbb{D}_{KL}\big(q_\phi(\gamma|x)||p_\theta(\gamma)\big),$$

where $\psi \in \mathbb{R}$ and $\lambda \in \mathbb{R}$ are the coefficients for the KL terms. This ELBO is approximated with Monte Carlo (MC) in practice, $\mathcal{L}(\theta, \phi; x)$:

$$\begin{aligned} &\frac{1}{N}\sum_{\gamma\sim q_\phi(\gamma|x)}^{N}\left[\frac{1}{M}\sum_{z\sim q_\phi(z|x,\gamma)}^{M}\log p_\theta(x|z)\right] - \\ &-\frac{\psi}{N}\sum_{\gamma\sim q_\phi(\gamma|x)}^{N}\Big[\mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma))\Big] - \\ &\qquad -\lambda\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)), \end{aligned} \quad (2)$$

where M and N are scalar numbers corresponding to a number of samples taken from $q_\phi(z|x,\gamma)$ and $q_\phi(\gamma|x)$ respectively. In this work, we set both M and N to 1. Similar to the vanilla VAE, the first term is the reconstruction, the second and the third KL terms control the distance between the posteriors and their corresponding priors. The parameters of the priors are fixed to some constant values (can be also thought as the hyperparameters) during the training. Also, see *Appendix* for ELBO derivation.

Control of Sparsity. The random variable γ_i, in our model, can be viewed as a "probabilistic switch" that determines how likely is for the ith dimension of z to be turned off. Intuitively, since for both generation and inference the latent code z is sampled from a Spike-and-Slab distribution with the mixture weights γ, $\gamma_i \to 1$ means z_i is drawn from a delta mass centered at $z_i = 0$. As the switch follows a Beta distribution $\gamma_i \sim Beta(\gamma_i; \alpha, \beta)$, we can select the parameters α and β to control the concentration of the probability mass on $\gamma_i \in [0, 1]$ interval.

There are three typical configurations of the (α, β) pair: (1) $\alpha < \beta$: density is shifted towards $\gamma_i = 0$ hence ith unit is likely to be on and dense representation is expected, (2) $\alpha = \beta$: the density is centered at $\gamma_i = 0.5$, and (3) $\alpha > \beta$: density is shifted towards $\gamma_i = 1$, hence the unit is likely to be off, leading to sparsity. The magnitude of these parameters also plays a role as it controls the spread and uni/bi-modal structure of the density.

4 Experiments

We conduct a set of experiments on three text classification corpora: Yelp (sentiment analysis - 5 classes) (Yang et al., 2017), DBpedia and Yahoo (topic classification - 14 and 10 classes respectively) (Zhang et al., 2015). First, we compare performance of the sparse latent representations with their dense counterpart on the text classification tasks (§4.2). Second, the stability of sparsification of HSVAE is compared with the state-of-the-art MAT-VAE (§4.3). Then, to better understand performance of our model on the downstream task, we examine the sparsity patterns (§4.4).

Remark. An integral part of the experiments is the analysis of the learned representations. In this sense, tasks that rely on understanding of semantics (e.g., GLUE (Wang et al., 2018)) or syntax (e.g., (Marvin and Linzen, 2018)) would be non-trivial to analyse due to their inherent complexity. We consider classification tasks because the distribution of words alone could be a good indicator of class labels. Given the unsupervised nature of the models, we explore if this surface-level distribution of words could be captured by the sparsity patterns in the learned representation.

4.1 Experimental Setup

4.1.1 Corpora Preprocessing

We use Yelp[5] as it is, without any additional preprocessing. As for DBpedia[6] and Yahoo[7], the preprocessing is as follows: (1) removing all non-ASCII characters, quotations marks, and hyperlinks, (2) tokenising with spaCy[8], (3) lower-case conversion for all tokens, then (4) **for each class** we randomly sample 10,000 sentences for the training corpus and 1,000 sentences for the test and validation respectively. The vocabulary size of the both corpora is reduced to the first 20,000 most frequent words.

4.1.2 Baselines and Models

To ground the performance of HSVAE we use 4 baselines: 1) VAE is a version of the vanilla VAE used in Higgins et al. (2017), 2) the same VAE model but the activation of μ and σ of $q_\phi(z|x)$ regularised by either L^1 (VAE$_{L^1}$) or L^2 (VAE$_{L^2}$) norms, 3) MAT-VAE is a VAE framework introduced by Mathieu et al. (2019) and 4) simple classifier which is simply a text encoder with a classifier on top of it. For all these models we use a GRU network (Cho et al., 2014) to encode and decode text sequences. We set the dimesnionality of the both encoder and the decoder GRU's to 512D and the dimensionality of the word embeddings is 256D. The decoder and the encoder share the word embeddings. To train the model we use the Adam optimiser (Kingma and Ba, 2014) with the learning rate: 0.0008.

BERT vs GRU Encoder. Inspired by Li et al. (2020b), we replace the GRU network used in VAE and HSVAE encoders with a pretrained BERT[9] (Devlin et al., 2019), while keeping the GRU decoder. We refer to these models as B-VAE and B-HSVAE, respectively. Also, we compare the

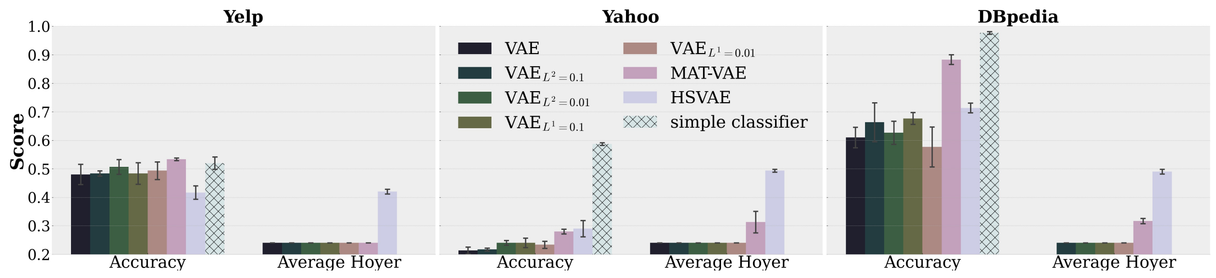
[5]`https://github.com/jxhe/vae-lagging-encoder/blob/master/prepare_data.py`.

[6]`https://github.com/srhrshr/torchDatasets/blob/master/dbpedia_csv.tar.gz`

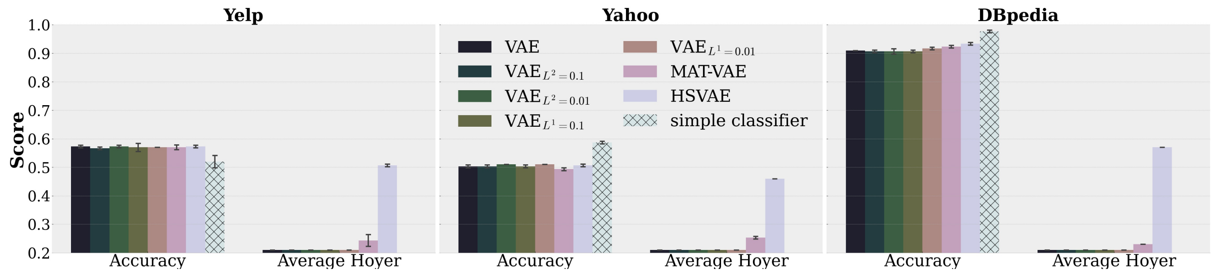
[7]`https://github.com/jxhe/vae-lagging-encoder/blob/master/prepare_data.py`.

[8]`https://spacy.io`

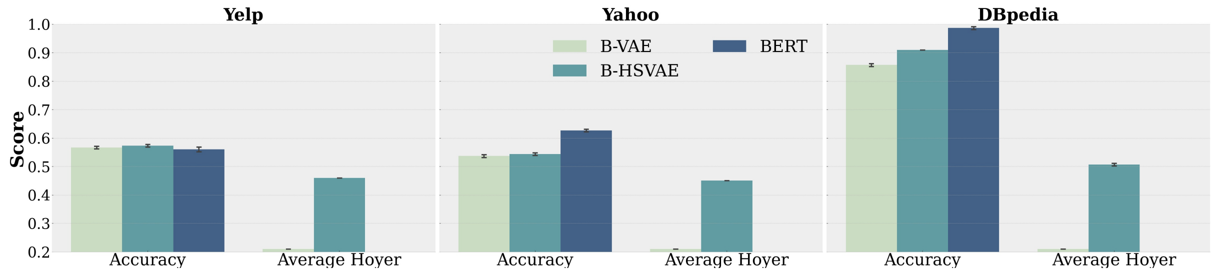
[9]After extracting features from a sequence with BERT, we then applying MLPs to extract features for the posterior distributions, as it is the case for the encoder with GRU network.



(a)



(b)



(c)

Figure 2: Classification Accuracy and Average Hoyer (higher means sparser z) for various VAE variants and the two baselines: simple classifier and BERT evaluated on Yelp, Yahoo or DBpedia test. The latent code of the VAEs is 32 D Figure (a) and 768 D Figures (b) and (c). Hoyer metric is not applicable to the simple classifier in the panels (a) and (b) and to the vanilla BERT model in the panel (c). The weights of the VAE encoders and BERT are **frozen** during the training of the classifiers. While the encoder of the simple classifier is updated during the training.

task performance of these VAE models with the plain pretrained base-BERT[10]. To train B-VAE and B-HSVAE, we use the Adam optimiser with the learning rate: 0.00008.

Dimensionality of *z*. We use the following two dimensions: 32D and 768D. Since, HSVAE and MAT-VAE induce sparse latent representations we want to make sure that they perform robustly regardless of the number of the dimensions.

KL-Collapse. None of the used VAE models is immune to the KL-collapse (Bowman et al., 2016) - when the KL term becomes zero and the decoder ignores the information provided by the encoder through z. To address this issue, in all the models, we put a scalar value $\psi, \lambda < 1$ on the KL terms of the VAE's objective function (He et al., 2019).

[10]https://huggingface.co/transformers/model_doc/bert.html

Coupling Encoder with Decoder. To connect the encoder with the decoder we concatenate the latent variable z, sampled from the posterior distribution, to word embeddings of the decoder at each time step (Prokhorov et al., 2019). Also, for GRU encoders we take the last hidden state to parameterise the posterior distribution. For BERT encoder, we take average pooling of all token's embeddings produced by the last layer of BERT.

4.1.3 Evaluation Metrics

Text Classification. To report the classification performance we use accuracy as a metric.

Sparsity. We measure Hoyer (Hurley and Rickard, 2009) on the representations of all data points in a corpus and report its average as our sparsity metric (Mathieu et al., 2019). Hoyer, in a nutshell, is ratio of the L^2 to L^1 norm, normalised by the number of dimensions. Higher indicates

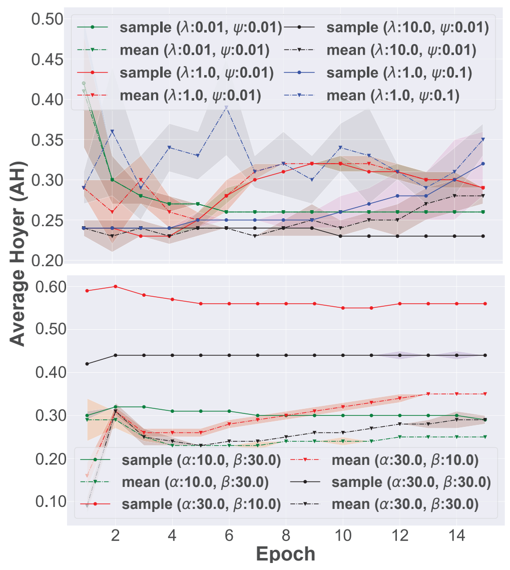


Figure 3: Average Hoyer (AH) on DBpedia corpus dev set for different parameterisations of Mathieu et al. (2019) (Top) vs. HSVAE (Bottom). Same is observed on Yelp and Yahoo (see *Appendix*). Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.

more sparsity. More specifically, to evaluate the average Hoyer, or as we refer to it as Average Hoyer (AH) in the experiments, either on a validation or test corpus we employ the following procedure. First, for each x_i in the corpus $\{x_1, ..., x_n\}$ we obtain its corresponding z_i by sampling it from a probabilistic encoder of a VAE model, such that for each x_i we sample one z_i: e.g. $x_1 \rightarrow z_1$. Then we normalise $\bar{z}_i = z_i/\sigma(z)$, where $z = \{z_1, ..., z_n\}$, and $\sigma(.)$ is the standard deviation. Finally, for each $\bar{z}_i$ we compute Hoyer as follows:

$$Hoyer(\bar{z}_i) = \frac{\sqrt{d} - ||\bar{z}_i||_1/||\bar{z}_i||_2}{\sqrt{d} - 1}, \qquad (3)$$

where d is the dimensionality of $\bar{z}_i$. To report the Hoyer for the whole corpus we compute the Average Hoyer $= \frac{1}{N}\sum_i^N Hoyer(\bar{z}_i)$, where N is the number of data points in a test or validation corpus.

4.2 Text Classification

Prior to use of a VAE encoder in the classification experiment, we pretrained it using the full VAE model with the corresponding VAE's objective function on one of the target corpus: Yelp, Yahoo or DBpedia. We compare performance of the sparse latent representations with their dense counterparts on the three text classification tasks (Figure 2). The classifier that we use comprises of the two dense layer of 32D each with the Leaky ReLU (Maas, 2013) activation function. To establish whether the performance gain or loss on the tasks is achieved thanks to the sparsity inductive bias, for all the VAE models and BERT we freeze the parameters of the encoder and only train the classifier which we put on top of the encoder. However, for the simple classifier model its text encoder is being trained together with the classifier. When the classifier, $p(y|x)$, is trained with a probabilistic VAE encoder we marginalise the latent variable(s). This is done for instance for HSVAE as,

$$p(y|x) = \int_{z,\gamma} p(y|z)q(z|x,\gamma)q_\phi(\gamma|x)dzd\gamma$$

We approximate the integral with MC by taking $K = 5$ samples from the probabilistic encoder both to train and to test the classifier: For each x_i in a batch $\{x_1, ..., x_p\}$:

1. sample K of $\gamma_{i,j}$ from $q_\phi(\gamma|x_i)$ i.e. a set of sampled γ's is $\{\gamma_{i,1}, ..., \gamma_{i,K}\}$
2. sample K of $z_{i,j}$ from $q_\phi(z|x_i, \gamma_{i,j})$ i.e. a set of sampled tuples of $z_{i,j}$ and $\gamma_{i,j}$ is $\{(z_{i,1}, \gamma_{i,1}), ..., (z_{i,K}, \gamma_{i,K})\}$ in other words for each $\gamma_{i,j}$ we sample only one $z_{i,j}$.

For the other VAEs the procedure is similar. With the MC approximation : $p(y|x) \approx 0.2 \times \sum_i^5 p(y|z_i)$.

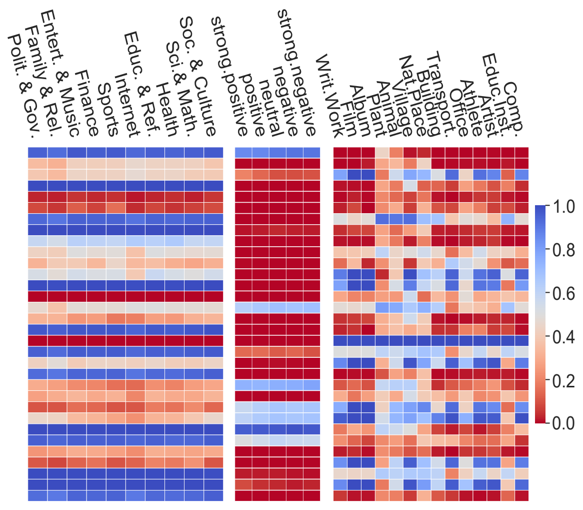
For a systematic comparison of various VAEs, we collate classification performance of VAEs with comparable reconstruction loss - which indicates how informative the latent code is for the decoder during reconstruction. In other words the reconstruction loss serves as an intrinsic metric. Thus, for an example, in Figure 2a, for the Yelp corpus all the VAE models have a similar reconstruction loss. The same applies to Figure 2b and Figure 2c.

Comparing the accuracy of the classifiers that are trained with the different latent representations i.e. sparse and dense (Figure2), shows that in general the performance of the sparse latent representations induced by HSVAE or MAT-VAE is on par with their dense latent counterparts inferred by the VAEs. However, the performance of HSVAE slightly lagging behind on the Yelp corpus when the dimensionality of the latent representation is 32D (Figure 2a). We put forward a hypothesis that may explain this in Section 4.4. Also, when the dimensionality of the latent representation is 32D, the accuracy of

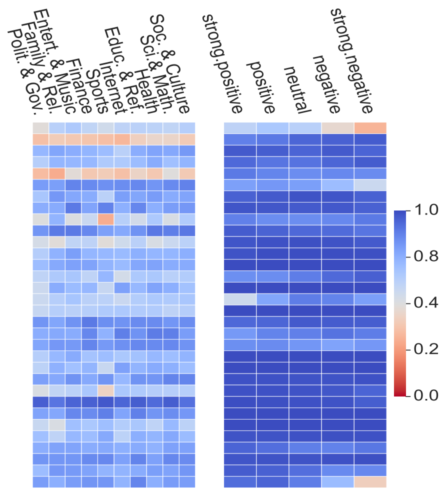
MAT-VAE is slightly better than of HSVAE, but this performance is reached at lower levels of sparsity. Additionally, we found that regularising the posterior parameters of the VAE model with either L^1 or L^2 norm, in some cases, helps to increase the classification accuracy, but does not reach AH higher than the vanilla VAE. Notably, the classification performance of all the VAE models becomes almost identical when the dimensionality of the latent space is increased from 32D to 768D, with HSVAE slightly outperforming all other VAEs on the DBpedia corpus (Figure 2b). We further elaborate on it in Section 4.4.

Use of BERT as an encoder, in our settings, only gives an improvement on the Yahoo corpus with B-HSVAE performing on par with B-VAE, but does not reach the classification accuracy of the plain BERT. We hypothesise that to reach the full potential of the use of a pretrained encoder in a VAE model one needs to pair it with a powerful decoder such as GPT-2 (Radford et al., 2019) as it is the case in the Li et al. (2020b) VAE model. Further exploration of this was beyond our compute resource.

Finally, one can observe that the simple classifier model performs on a par (in Figure 2a) or even worse (Figure 2b) than the VAE models on the Yelp corpus. Putting it into the context that the VAE encoders are not being trained with a supervision signal while the encoder of the simple classifier is, we speculate that this can be explained by the discussion put forward in Valpola (2014). A classifier in nature tries to remove all the information that is not relevant to the supervision signal, while an autoencoder tries to preserve as much as possible information in the latent code in order to reconstruct the original input data reliably. Thus, if the distribution of class related words in a text alone (see §4.4.1) is not indicative enough of a class then the classifier may perform poorly. In our case, we hypothesise that the VAE models capture some additional information other than class distribution of words in text that allows it to better discriminate the classes. For example, some class may have shorter sentences, on average, than the sentences presented in the other classes. This may provide an additional bias that allows the VAE models to discriminate sentences from this class from the sentences from the other classes. Thus, with this additional bias VAEs can perform better than the simple classifier. We leave this investigation for a future work.



(a)



(b)

Figure 4: Heat maps of γ_{class} (Section §4.4). (a) γ_{class} of 32D - from left to right: Yahoo, Yelp, DBpedia. (b) contiguous 32D out of 768D of γ_{class} - from left to right: Yahoo, Yelp.

4.3 Representation Sparsity

In Figure 7 we compare HSVAE with MAT-VAE. We report AH both on the mean and samples from the posterior distributions. As illustrated, MAT-VAE struggles to achieve steady and consistent AH regardless of the configurations of its hyperparameters (ψ, λ). However, HSVAE stably controls the level of sparsity with α and β parameters, a positive effect of its more flexible posterior distribution and the learnable distribution over γ.

4.4 Can Sparsity Patterns Encode Classes?

In order to identify pertinent features, the unsupervised representation learning models are typically trained/fine-tuned on corpora that are closely related to the downstream task. As such, without a supervisory signal, the model can only rely on the distribution of words in a text in order to identify these relevant features for the task. Ideally, compared to their dense counterparts, an unsupervised sparsification model such as HSVAE could result in performance improvement on downstream tasks if they capture the task-related features and discard the noisy features. However, if the sparsification model fail to capture the task related signal in its sparsity pattern; it can hurt the performance of the model on the downstream task as the task-related information can be removed. In what follows we investigate this direction by analysing the sparsity patterns and relate this analysis to the classification performance of the model (§4.2).

Analysis of γ. We hypothesise that if γ captures a class of a sentence then the sentences that belong to the same class should have a similar sparsity patterns in γ. To obtain a class specific γ_{class}, first, for each sentence x we obtain the mean of the posterior distribution: $q_\phi(\gamma|x)$ and we denote it as $\mu_{\gamma(x)}$. Then we binarise the mean such as $\mu^b_{\gamma(x)} = \text{Binarise}(\mu_{\gamma(x)})$, where $\text{Binarise}(\cdot)$ is defined as: 0 if $\mu_{\gamma(x)} < 0.5$ and 1 otherwise. Finally, for each class we average its $\mu^b_{\gamma(x)}$ vectors to obtain a single vector that represent this class: $\gamma_{class} = \frac{1}{M} \sum_{x \in class} \mu^b_{\gamma(x)}$, where M is a number of sentences in the class. The averaging removes the information that differentiate these sentences, while preserving the class information that is shared among them. A similar approach was also used in Mathieu et al. (2019).

Figure 4 reports the magnitudes of the γ_{class} vectors as heat maps for the three corpora. One would expect that γ_{class} of different classes should differ. For 32D γ_{class} (Figure 4a) this is the case when HSVAE is trained on the DBpedia and Yahoo but not on Yelp. Taking into account the unsupervised nature of these models, this difference is echoing the distribution of words in the classes, which is more distinct in DBpedia and Yahoo, but not in Yelp (see §4.4.1). We also hypothesis that this observation can explain inferior performance of the model on the Yelp corpus (Figure 2a).

In contrast, for γ_{class} in 768D (Figure 4b) one can observe that the different classes have different activation patterns even when HSVAE is trained on the Yelp corpus.[11] Also, the distributedness of the activation patterns now becomes more apparent when HSVAE is trained on the Yahoo corpus. This observation is also related to the distribution of words in the text (further elaborated in §4.4.1).

Intuitively, to reconstruct a sentence a VAE model first captures aspect of data that are the most conducive for reconstruction error reduction (Burgess et al., 2018). Therefore, given the limited dimensionality of the latent vector, the model will prioritised aspects of data during encoding. As such, if the information such as sentence class is not strongly presented in the corpus the model could potentially ignore it during encoding. However, when the dimensionality of the latent space is increased, the model has more capacity to represent various aspects of data that may otherwise be ignored in the smaller dimensionality. We speculate this could explain the presence of distributedness of γ_{class} on Yelp for 768D as opposed to 32D, which also translates into matching the task performance of its dense counterpart (Figure 2b).

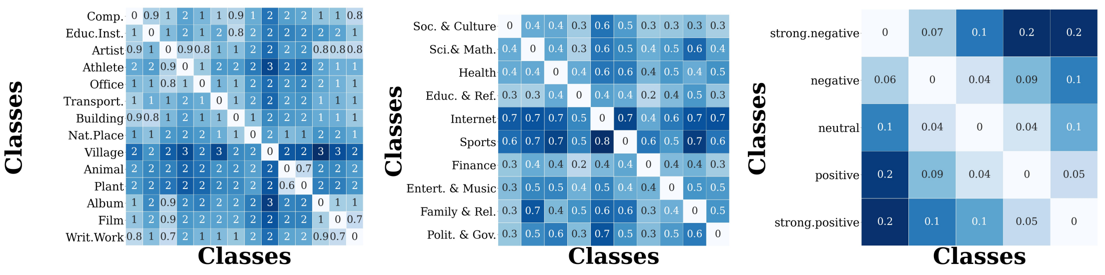
4.4.1 Class Kullback–Leibler Divergence

The question that has yet not been addressed is why in some cases the HSVAE model is more successful at capturing the class distribution when trained on DBpedia compared to Yelp. We previously hypothesised that the reason for this can be a word distribution in a text. To empirically test our hypothesis, we calculate the add-1 smoothed probabilities of words in the classes and measure the pairwise KL divergence across them. The magnitudes of the pairwise KL divergences are shown in Figure 5. As demonstrated, the magnitude of the KL divergence is the largest for DBpedia and smallest for Yelp. This indicates that separating classes in Yelp would rely on more subtle aspects of data, whereas surface-level cues are more present in DBpedia and allow for an easier discrimination.

5 Related Work

Learning sparse representations of data can be dated back to Olshausen and Field (1996). This work motivates encoding of images in sparse linear codes for its biological plausibility and efficiency. It was later argued by Bengio (2009) that compared to the dimensionality reduction approaches, sparsity is a more efficient method for representation learning on vectors with fixed dimensionality.

[11] In Figure 4b we only show 32D out of 768D. This is one of the subsets of the 768 dimensions where the distributedness is present. It is not unique and the distributedness is also present in other dimensions of the 768D code.



(a) (b) (c)

Figure 5: Experimental results for KL between classes on the three corpora: DBpedia (a), Yahoo (b) and Yelp (c).

Representation Sparsity. In NLP, learning sparse representations has been explored for various units of text with most of the focus placed on sparse representation of words. As the earliest work that moved in this direction, Murphy et al. (2012) looked into sparse representations for ease of analysis, performance, and being more cognitively plausible. This idea was further developed by many other researchers (Faruqui and Dyer, 2015; Yogatama et al., 2015; Faruqui et al., 2015; Sun et al., 2016; Subramanian et al., 2018; Arora et al., 2018; Li and Hao, 2019). Sparsification of the large units of text (i.e., sentences) has not received a lot of attention, perhaps due to inherent complexity of sentence/phrase representations: i.e., encoding and analysing syntactic and semantic information in a sentence embedding is rather a non-trivial task. To the best of our knowledge, the only model that sparsifies sentence emebeddings is introduced by Trifonov et al. (2018). The authors introduced a Seq2Seq model (Sutskever et al., 2014) with the Sparsemax layer (Martins and Astudillo, 2016) between the encoder and the decoder which induces sparse latent codes of text. This layer allows to learn codes that can be easier to analyse compared to their dense counterparts, but it is limited to modelling the categorical distribution. Thus restricts a type a sentence representations that can be learned.

VAE-based Representation Sparsity. VAE-based sentence representation learning has shown superior properties compared to their deterministic counterparts on tasks such as text generation (Bowman et al., 2016), Semantic Textual Similarity (Li et al., 2020a) and other wide range of language tasks (Li et al., 2020b). While a handful of VAE-based sparsification methods have been proposed recently Mathieu et al. (2019) (MAT), Tonolini et al. (2019) (TON), they have been only evaluated on image domain. We summarise the similarity and key differences with HSVAE model:

PRIOR AND POSTERIOR. All three frameworks use the Spike-and-Slab distribution to construct the prior on z. While the posterior distribution in MAT remains as a Gaussian, both TON and HSVAE opt for Spike-and-Slab. However, TON controls the sparsity level in an indirect way via "pseudo data" (Tomczak and Welling, 2018) used in prior, whereas HSVAE's probabilistic treatment of γ enables direct control on the target sparsity level.

OBJECTIVE. HSVAE is trained with a principled ELBO (eq. 3), while the other two add additional regularisers to the ELBO of VAE (eq. 1). For instance, MAT add a *maximum mean discrepancy* (MMD) divergence between z's aggregated posterior and prior $\mathrm{MMD}(q_\phi(z), p_\theta(z))$ and include scalar ψ and λ weights to the KL and MMD term, respectively, see *Appendix*.

Model Sparsity. Concurrent to the widespread use of large models such as Transformers (Vaswani et al., 2017) in NLP, sparsification of these models is also becoming popular (Zhang et al., 2020; Zhao et al., 2019; Correia et al., 2019; Ye et al., 2019; Child et al., 2019). The most common approach to sparsify a Transformer is to reduce a number of connection between the words/tokens in the self attention kernel e.g. Correia et al. (2019). However, these approaches still learn dense continuous representations of token/word/sentence embeddings.

6 Conclusion

We provided an objective analysis of several unsupervised sparsification frameworks based on VAEs, both in terms of the impact on downstream tasks and the level of sparsity achieved. Also, we presented a novel VAE model - Hierarchical Sparse Variational Autoencoder (HSVAE), outperforming existing SOTA model (Mathieu et al., 2019). Ideally, sparse representations should be capable of encoding the underlying characteristics of a corpus (e.g. class), in activation patterns as shown to be the case for HSVAE. Moreover, using the text classification corpora as a testbed, we established how statistical properties of a corpus such as word distribution in a class affect the ability of learned sparse codes to represent task-related information.

Moving forward, HSVAE model along with the analysis provided in this paper can serve as a good basis for the design of sparse models that induce continuous sparse vectors of text. For example, a potential extension of HSVAE could be an incorporation of explicit linguistic biases into the learned representations with the group sparsity (Yogatama et al., 2015). Furthermore, as we discussed in Section 5, sparsity found its application in the Transformers, but it, mainly, has been used to reduce the number of connection between the words/tokens. With the HSVAE framework one can also learn sparse continuous representations of token/word/sentence embeddings.

Acknowledgments

The first author would like to thank Yi Zhu for providing his feedback on the earlier vesrsion of the paper. The authors, also, would like to thank the three anonymous reviewers for their helpful suggestions.

References

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2018. Linear algebraic structure of word senses, with applications to polysemy. *Transactions of the Association for Computational Linguistics*, 6:483–495.

Vikash Balasubramanian, Ivan Kobyzev, Hareesh Bahuleyan, Ilya Shapiro, and Olga Vechtomova. 2020. Polarized-vae: Proximity based disentangled representation learning for text generation. *arXiv preprint arXiv:2004.10809*.

Gabriel Barello, Adam S. Charles, and Jonathan W. Pillow. 2018. Sparse-coding variational auto-encoders. *bioRxiv*.

Yoshua Bengio. 2009. Learning deep architectures for ai. *Found. Trends Mach. Learn.*, 2(1):1–127.

Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. 2013. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *CoNLL*.

Christopher P. Burgess, Irina Higgins, Arka Pal, Loïc Matthey, Nick Watters, Guillaume Desjardins, and Alexander Lerchner. 2018. Understanding disentangling in β-vae. *CoRR*, abs/1804.03599.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *CoRR*, abs/1406.1078.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. 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 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Manaal Faruqui and Chris Dyer. 2015. Non-distributional word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 464–469, Beijing, China. Association for Computational Linguistics.

Manaal Faruqui, Yulia Tsvetkov, Dani Yogatama, Chris Dyer, and Noah A. Smith. 2015. Sparse overcomplete word vector representations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1491–1500, Beijing, China. Association for Computational Linguistics.

Michael Figurnov, Shakir Mohamed, and Andriy Mnih. 2018. Implicit reparameterization gradients. In *Proceedings of the 32nd International Conference on*

Neural Information Processing Systems, NIPS'18, page 439–450, Red Hook, NY, USA. Curran Associates Inc.

Junxian He, Daniel Spokoyny, Graham Neubig, and Taylor Berg-Kirkpatrick. 2019. Lagging inference networks and posterior collapse in variational autoencoders. In *Proceedings of ICLR*.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, Toulon, France.

N. Hurley and S. Rickard. 2009. Comparing measures of sparsity. *IEEE Transactions on Information Theory*, 55(10):4723–4741.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. 2020a. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9119–9130, Online. Association for Computational Linguistics.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020b. Optimus: Organizing sentences via pre-trained modeling of a latent space.

Wenye Li and Senyue Hao. 2019. Sparse lifting of dense vectors: Unifying word and sentence representations. *CoRR*, abs/1911.01625.

Andrew L. Maas. 2013. Rectifier nonlinearities improve neural network acoustic models.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2016. The concrete distribution: A continuous relaxation of discrete random variables. *International Conference on Learning Representations, ICLR*.

Andre Martins and Ramon Astudillo. 2016. From softmax to sparsemax: A sparse model of attention and multi-label classification. In *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1614–1623, New York, New York, USA. PMLR.

Rebecca Marvin and Tal Linzen. 2018. Targeted syntactic evaluation of language models. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, Brussels, Belgium. Association for Computational Linguistics.

Emile Mathieu, Tom Rainforth, N Siddharth, and Yee Whye Teh. 2019. Disentangling disentanglement in variational autoencoders. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 4402–4412, Long Beach, California, USA. PMLR.

Brian Murphy, Partha Talukdar, and Tom Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *Proceedings of COLING 2012*, pages 1933–1950, Mumbai, India. The COLING 2012 Organizing Committee.

Bruno Olshausen and David Field. 1996. Emergence of simple-cell receptive field properties by learning a sparse code for natural images. *Nature*, 381:607–9.

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, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Anant Subramanian, Danish Pruthi, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Eduard H. Hovy. 2018. SPINE: sparse interpretable neural embeddings. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence, (AAAI-18), the 30th innovative Applications of Artificial Intelligence (IAAI-18), and the 8th AAAI Symposium on Educational Advances in Artificial Intelligence (EAAI-18), New Orleans, Louisiana, USA, February 2-7, 2018*, pages 4921–4928. AAAI Press.

Fei Sun, Jiafeng Guo, Yanyan Lan, Jun Xu, and Xueqi Cheng. 2016. Sparse word embeddings using l1 regularized online learning. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, pages 2915–2921. IJCAI/AAAI Press.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Jakub M. Tomczak and Max Welling. 2018. Vae with a vampprior. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 1214–1223*.

Francesco Tonolini, Bjorn Sand Jensen, and Roderick Murray-Smith. 2019. Variational sparse coding. In *Proceedings of the Thirty-Fifth Conference on Uncertainty in Artificial Intelligence (UAI)*.

Valentin Trifonov, Octavian-Eugen Ganea, Anna Potapenko, and Thomas Hofmann. 2018. Learning and evaluating sparse interpretable sentence embeddings. In *Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, Brussels, Belgium, November 1, 2018*, pages 200–210. Association for Computational Linguistics.

H. Valpola. 2014. From neural pca to deep unsupervised learning. *ArXiv*, abs/1411.7783.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Peng Xu, Jackie Chi Kit Cheung, and Yanshuai Cao. 2019. On variational learning of controllable representations for text without supervision. *arXiv preprint arXiv:1905.11975*.

Zichao Yang, Zhiting Hu, Ruslan Salakhutdinov, and Taylor Berg-Kirkpatrick. 2017. Improved variational autoencoders for text modeling using dilated convolutions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3881–3890, International Convention Centre, Sydney, Australia. PMLR.

Zihao Ye, Qipeng Guo, Quan Gan, Xipeng Qiu, and Zheng Zhang. 2019. Bp-transformer: Modelling long-range context via binary partitioning.

Serena Yeung, Anitha Kannan, Yann Dauphin, and Li Fei-Fei. 2017. Tackling over-pruning in variational autoencoders. *International Conference on Machine Learning: Workshop on Principled Approaches to Deep Learning*.

Dani Yogatama, Manaal Faruqui, Chris Dyer, and Noah A. Smith. 2015. Learning word representations with hierarchical sparse coding. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37, pages 87–96. JMLR.org.

Biao Zhang, Ivan Titov, and Rico Sennrich. 2020. On sparsifying encoder outputs in sequence-to-sequence models.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, page 649–657, Cambridge, MA, USA. MIT Press.

Guangxiang Zhao, Junyang Lin, Zhiyuan Zhang, Xuancheng Ren, Qi Su, and Xu Sun. 2019. Explicit sparse transformer: Concentrated attention through explicit selection.

A Derivations of ELBO

Starting from the $\mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x))$, we derive the Evidence Lower Bound (ELBO) as follows:

$$\mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x)) = \int\limits_{z,\gamma} dz d\gamma\, q_\phi(z,\gamma|x) \log \frac{q_\phi(z,\gamma|x)}{p_\theta(z,\gamma|x)}, \tag{4}$$

after rearranging terms in equation 4 we can obtain:

$$\log p_\theta(x) - \mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x)) = \underbrace{\int\limits_{z,\gamma} dz d\gamma\, q_\phi(z,\gamma|x) \log \frac{p_\theta(z,\gamma,x)}{q_\phi(z,\gamma|x)},}_{\text{ELBO}} \tag{5}$$

Based on the independence assumption that we make in our graphical model (Figure 1) the generative model factorises as: $p_\theta(z,\gamma,x) = p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)$ and the inference model factorises as: $q_\phi(z,\gamma|x) = q_\phi(z|\gamma,x)q_\phi(\gamma|x)$. Therefore, we can rewrite the ELBO as follows:

$$\int\limits_{z,\gamma} dz d\gamma\, q_\phi(z|\gamma,x)q_\phi(\gamma|x) \log \frac{p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)}{q_\phi(z|\gamma,x)q_\phi(\gamma|x)}, \tag{6}$$

We can further rewrite the ELBO as a sum of the three separate terms. Where the first term is:

$$\begin{aligned} &\int\limits_{z,\gamma} dz d\gamma\, q_\phi(z|x,\gamma)q_\phi(\gamma|x) \log p_\theta(x|z) \\ &\int\limits_{\gamma} d\gamma\, q_\phi(\gamma|x) \int\limits_{z} dz\, q_\phi(z|x,\gamma) \log p_\theta(x|z) \therefore \\ &\left\langle \int\limits_{z} dz\, q_\phi(z|x,\gamma) \log p_\theta(x|z) \right\rangle_{q_\phi(\gamma|x)} \therefore \end{aligned} \tag{7}$$

The second term is:

$$\int_{z,\gamma} dzd\gamma\, q_\phi(z|x,\gamma)q_\phi(\gamma|x)[\log q_\phi(z|x,\gamma) - \log p_\theta(z|\gamma)]$$
$$\left\langle \int_z dz\, q_\phi(z|x,\gamma)[\log q_\phi(z|x,\gamma) - \log p_\theta(z|\gamma)] \right\rangle_{q_\phi(\gamma|x)} \therefore$$
$$\left\langle \mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma)) \right\rangle_{q_\phi(\gamma|x)} \therefore \tag{8}$$

Finally, the third term is:

$$\int_{z,\gamma} dzd\gamma\, q_\phi(z|x,\gamma)q_\phi(\gamma|x)[\log q_\phi(\gamma|x) - \log p_\theta(\gamma)]$$
$$\int_\gamma d\gamma\, q_\phi(\gamma|x)[\log q_\theta(\gamma|x) - \log p_\theta(\gamma)] \underbrace{\int_z dz\, q_\phi(z|x,\gamma)}_{\text{sums to 1 for each}:\gamma} \therefore$$
$$\int_\gamma d\gamma\, q_\phi(\gamma|x)[\log q_\phi(\gamma|x) - \log p_\theta(\gamma)] \therefore$$
$$\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)) \therefore \tag{9}$$

Collecting all the three terms into the single ELBO:

$$\left\langle \int_z dz\, q_\phi(z|x,\gamma) \log p_\theta(x|z) \right\rangle_{q_\phi(\gamma|x)} -$$
$$- \left\langle \mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma)) \right\rangle_{q_\phi(\gamma|x)} - \tag{10}$$
$$- \mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)),$$

B Objective Functions of Mathieu et al. (2019) and Tonolini et al. (2019) Models

The objective function of Mathieu et al. (2019) is:

$$\left\langle \log p_\theta(x|z) \right\rangle_{q_\phi(z|x)} - \psi KL(q_\phi(z|x)||p_\theta(z)) -$$
$$- \lambda \mathbb{D}(q_\phi(z), p_\theta(z)),$$

where ψ and λ are the scalar weight on the terms and Tonolini et al. (2019) is:

$$\left\langle \log p_\theta(x|z) \right\rangle_{q_\phi(z|x)} - KL(q_\phi(z|x)||q_\phi(z|x_u)) -$$
$$- J \times \mathbb{D}_{KL}\big(\bar{\gamma}_u||\alpha\big),$$

where J is the dimensionality of the latent variable z, x_u is a learnable pseudo-input (Tomczak and Welling, 2018) and α is prior sparsity.

C Deriving Marginal of (Univariate) Spike-and-Slab Prior

We derive the Spike-and-Slab distribution by integrating out the index component which is distributed as a Bernoulli variable. This result is quite well-known in machine learning, however for the ease of the reader we present it here as a quick reference.

The derivation: assume 1) $\pi \sim p(\pi;\gamma)$ is a $Bernoulli(\gamma)$ and 2) $p(z|\pi) = (1-\pi) \times p_1(z) + \pi \times p_2(z)$, where $p_1(z) \sim \mathcal{N}(z;0,1)$ and $p_2(z) \sim \mathcal{N}(z;0,\sigma \to 0)$ is a Spike-and-Slab model. The the marginal Spike-and-Slab prior over z can be obtained in the following way:

$$p(z;\gamma) = \sum_{i=0}^{1} p(z|\pi = i)p(\pi = i;\gamma)$$
$$p(z|\pi = 0)p(\pi = 0;\gamma) + p(z|\pi = 1)p(\pi = 1;\gamma) \therefore$$
$$[(1-0) \times p_1(z) + 0 \times p_2(z)]p(\pi = 0;\gamma) +$$
$$+ [(1-1) \times p_1(z) + 1 \times p_2(z)]p(\pi = 1;\gamma) \therefore$$

Expanding brackets:

$$p_1(z)p(\pi = 0;\gamma) + p_2(z)p(\pi = 1;\gamma) \therefore$$
$$\mathcal{N}(z;0,1)p(\pi = 0;\gamma) + \mathcal{N}(z;0,\sigma \to 0)p(\pi = 1;\gamma) \therefore$$
$$(1-\gamma)\mathcal{N}(z;0,1) + \gamma\mathcal{N}(z;0,\sigma \to 0) \therefore$$

Therefore,

$$p(z;\gamma) = (1-\gamma)\mathcal{N}(z;0,1) + \gamma\mathcal{N}(z;0,\sigma \to 0).$$

D End-to-end Differentiable

Sampling a value from the Spike-and-Slab posterior distribution $q(z|x,\gamma)$ is a two step process. First a spike or slab component is sampled which is a binary decision, we use Binary Concrete distribution (Maddison et al., 2016) to make this sampling step end-to-end differentiable. Then the value is sampled from the corresponding component, for this we employ the reparameterization trick (Kingma and Welling, 2014). Also, samples from the Beta distribution are pathwise differentiable (Figurnov et al., 2018).

E Hoyer

This section reports Average Hoyer, for the two corpora Yelp and Yahoo, both on the mean and samples from the posterior distributions of the HSVAE and MAT-VAE models.

E.1 MAT-VAE

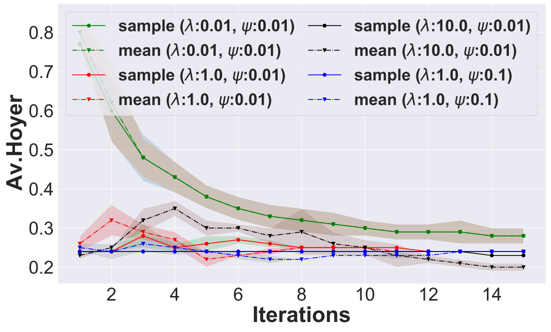


Figure 6: Average Hoyer (Av.Hoyer) on Yelp corpus dev set for MAT-VAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.

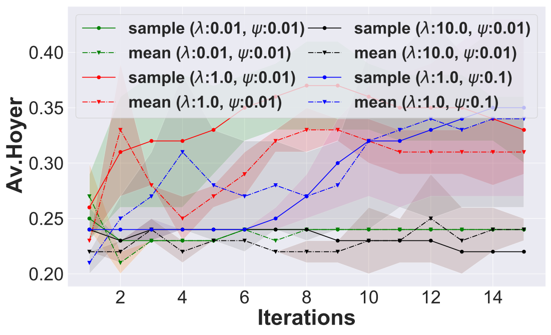


Figure 7: Average Hoyer (Av.Hoyer) on Yahoo corpus dev set for MAT-VAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation.

E.2 HSVAE

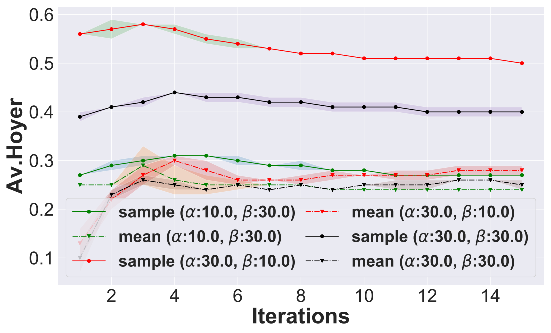


Figure 8: Average Hoyer (Av.Hoyer) on Yelp corpus dev set for HSVAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.

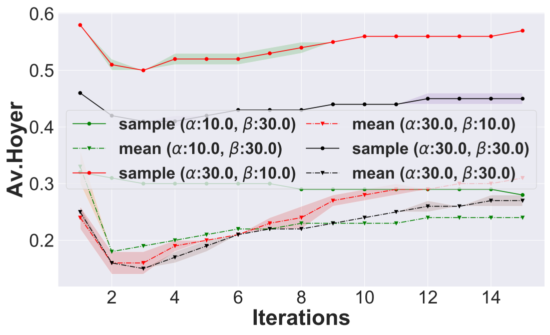


Figure 9: Average Hoyer (Av.Hoyer) on Yahoo corpus dev set for HSVAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation. The dimensionality of the latent variable of the models is 32D.

F Hardware

Please refer to Table 1 for the hardware that we use.

hardware	specification
CPU	Intel® Xeon E5-2670V3, 12-cores, 24-threads
GPU	NVIDIA® TITAN RTX™ (24 GB) x 1
RAM	CORSAIR® Vengeance LPX DDR4 2400 MHz (8 GB) x 4

Table 1: Computing infrastructure.

G Datasets

	Yelp	DBpedia	Yahoo
# sent. (train corpus)	100K	140K	100K
# sent. (valid corpus)	10K	14K	10K
# sent. (test corpus)	10K	14K	10K
vocabulary size	19,997	20K	20K
min sent. length.	20	1	5
av. sent. length.	96	35	12
max. sent. length.	200	60	30
# classes	5	14	10
# sent. in each class (train/test corpus)	20K/2K	10K/1K	10K/1K

Table 2: Statistics of corpora. Vocabulary size excludes the ⟨pad ⟩and ⟨EOS ⟩symbols.