Hierarchical Sparse Variational Autoencoder for Text Encoding

Victor Prokhorov[♣], **Yingzhen Li**[♠], **Ehsan Shareghi**[◇,♣], **Nigel Collier**[♣]
[♣] Language Technology Lab, University of Cambridge
[♠]Microsoft Research Cambridge ,[◇]University College London
[♣]`{vp361, nhc30}@cam.ac.uk`,[♠]`Yingzhen.Li@microsoft.com`
[◇]`e.shareghi@ucl.ac.uk`

Abstract

In this paper we focus on unsupervised representation learning and propose a novel framework, Hierarchical Sparse Variational Autoencoder (HSVAE), that imposes sparsity on sentence representations via direct optimisation of Evidence Lower Bound (ELBO). Our experimental results illustrate that HSVAE is flexible and adapts nicely to the underlying characteristics of the corpus which is reflected by the level of sparsity and its distributional patterns.

1 Introduction

Representation learning has been the key fabric of the success stories in modern days Natural Language Processing (NLP). On the one hand, it is well-documented that each data point utilises a different sub-space of this high dimensional representation space (Coates and Ng, 2011; Bengio et al., 2013; Burgess et al., 2018; Tonolini et al., 2019), reminiscent of cognitive findings that humans use different subsets of cognitive features depending on concepts (Vinson and Vigliocco, 2008) (and references therein). On the other hand, sparse representations are of specific interest as a pathway towards interpretability (Murphy et al., 2012; Lipton, 2016).

These observations encouraged 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; Trifonov et al., 2018) 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 and restrictive assumption by forcing a preset number of active dimensions.

In parallel, Variational Autoencoder (VAE), Figure 1 (left), (Kingma and Welling, 2014) and its modifications (Burgess et al., 2018; Razavi et al., 2019) have been shown to be effective in capturing semantic closeness of sentences via the learned latent representation space (Bowman et al., 2016; Prokhorov et al., 2019; Xu et al., 2019; Balasubramanian et al., 2020). Furthermore, methods have been developed for encouraging sparsity in VAEs via learning a deterministic selection variable (Yeung et al., 2017) or through sparse priors (Barello et al., 2018; Mathieu et al., 2019; Tonolini et al., 2019). However, these have not been tested on text.

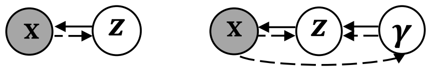


Figure 1: Graphical Models of VAE (left) and HSVAE (right). Solid and dashed lines represent generative and inference paths, respectively.

To bridge this gap, we propose a Hierarchical Sparse Variation Autoencoder (HSVAE) [1], Figure 1 (right), a fully probabilistic framework trained by direct optimisation of ELBO. Our experimental results on various corpora illustrate that HSVAE adapts to the distributional properties of data. This is reflected by the amount of sparsity achieved and its distributional patterns in the learned representations.

Compared to Mathieu et al. (2019), which is the most relevant work to ours, we show the success they achieved on the image domain does not transfer to text, while HSVAE successfully induces sparse sentence representations: we attribute this to (a) the lack of flexibility in their Gaussian inference network compared to the bi-modal spike-and-slab posterior of HSVAE, and (b) the use of pre-fixed weights on spike components which in our case is inferred per each input instance. To the best of our knowledge, HSVAE is the first VAE-based framework for learning sparse text encodings.

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

2 Background

Variational Autoencoder (VAE). Given an input x, VAEs 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:

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

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 $\sigma \to 0$:

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

where i denotes the ith dimension of z.

3 Hierarchical Sparse VAE (HSVAE)

We propose the hierarchical sparse VAE (HSVAE) to learn sparse latent codes automatically. In detail, we treat the mixture weights $\gamma = (\gamma_1, ..., \gamma_D)$ as a random variable and assign a factorised Beta prior $p_\theta(\gamma_i) = \text{Beta}(\alpha, \beta)$ 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 dimensions. 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. The graphical model of HSVAE is shown in Figure 1 (right) with a comparison to the vanilla VAE.

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

$$\Big\langle \log p_\theta(x|z) \Big\rangle_{q_\phi(z,\gamma|x)} - \psi \Big\langle \mathbb{D}_{KL}(q_\phi(z|\gamma,x)||p_\theta(z|\gamma)) \Big\rangle_{q_\phi(\gamma|x)} \quad (2)$$
$$- \lambda \mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma))$$

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. 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. See Appendix A for details of ELBO derivations and the MC approximation.

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.

Related Work. Of particular relevance to our model are the VAE-based frameworks of Mathieu et al. (2019) (MAT), and Tonolini et al. (2019) (TON). We summarise the similarity and key differences:

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 more direct control on the target sparsity level.

OBJECTIVE. HSVAE is trained with a principled ELBO (eq. 2), 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 MMD($q_\phi(z), p_\theta(z)$) and include scalar ψ and λ weights to the KL and MMD term, respectively. See Appendix B for the full objective functions of MAT and TON.

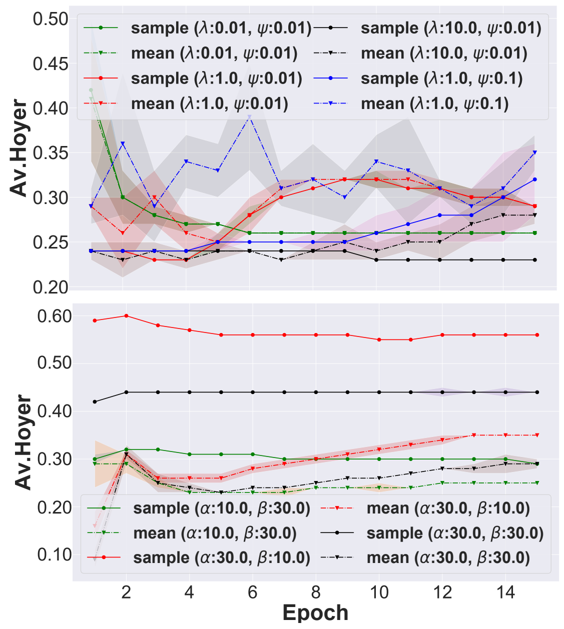


Figure 2: Average Hoyer (Av.Hoyer) on DBpedia corpus dev set for different parameterisations of Mathieu et al. (2019) (Top) vs. HSVAE (Bottom). Same is observed on Yelp. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation.

4 Experiments

We conduct a set of experiments on two text classification corpora: Yelp (sentiment analysis - 5 classes) (Yang et al., 2017) and DBpedia (topic classification - 14 classes) (Zhang et al., 2015) to investigate whether: (1) HSVAE learns sparse and distributed representations that are reflective of the underlying characteristics of the corpus, and (2) the quality of the sparse latent representations is comparable to their dense counterpart on text classification tasks. Our representations are 32D. For details on data statistics and configurations of the models and optimiser see Appendix H.

4.1 Sparsity

In Figure 5 we compare our model against Mathieu et al. (2019) using their suggested sparsity metric, Av.Hoyer, which is an average Hoyer (Hurley and Rickard, 2009) over the number of data points of a dataset (dev set).[2] Hoyer of a vector, in a nutshell, is ratio of the L^2 norm to L^1 norm, normalised by the number of dimensions. Higher Hoyer means more sparsity. We report Hoyer both, on the mean of the posterior distributions of $q_\phi(z|x)$ and $q_\phi(z|x,\gamma)$ and the samples from these distributions. As illustrated, their model struggles to achieve steady and consistent Av.Hoyer regardless of the configurations of ψ and λ. However, HSVAE stably controls the level of sparsity with α and β parameters, a positive of its more flexible posterior distribution and the learnable distribution over γ.

[2] Similar results are observed on the test set (Figure 3 (a)).

4.2 Distributedness

First, we investigate whether the sparsity patterns are distributed without analysing what properties they encode. Figure 3 (a) reports Av.Hoyer of the test corpus and the Hoyer computed on the average of zs (Agg.Hoyer). Intuitively, the sparsity patterns are not distributed if the Agg.Hoyer and Av.Hoyer values are roughly the same, indicating the same dimensions are being on/off for all the sentences. For both corpora, the Agg.Hoyer is much lower than the Av.Hoyer, especially for ($\alpha = 30, \beta = 10$) where sparsity is encouraged. Next, we investigate if class information is captured by these patterns.

4.2.1 Can Sparsity Patterns Encode Classes?

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.[3] We use the mean ($\mu_{\gamma(x)}$) of the $q_\phi(\gamma|x)$ distribution as a γ vector of a sentence x. To get an encoding of a class - γ_{class} - we average all $\mu_{\gamma(x)}$ of that class (see Appendix E). The averaging removes the information that differentiate these sentences, while preserving the information that are shared among them - the class. Figure 4 (a) reports the magnitudes of the γ_{class} vectors as a heat map: darker colours indicate lower magnitudes of γ_{class} denoting the dimension being active, while brighter colours show the opposite. One would expect that γ_{class} of different classes should differ.

As demonstrated, this is the case when the HSVAE$_{\alpha=30,\beta=10}$ is trained on the DBpedia but not on Yelp. Taking into account that our model is fully unsupervised, an explanation for this can be that a distribution of words in the classes of DBpedia is more distinct than in Yelp. To verify this, we calculate the add-1 smoothed probabilities of words in the classes and measure the pairwise KL divergence across the classes. The magnitudes of the pairwise KL divergences in Figure 4 (b) confirms our hypothesis. The sparsity patterns of HSVAE$_{\alpha=10,\beta=30}$ for both corpora are uniform, this is expected as z vectors are more dense (i.e.,

[3]We expect that only a subset of the dimensions encoding class label information are similar.

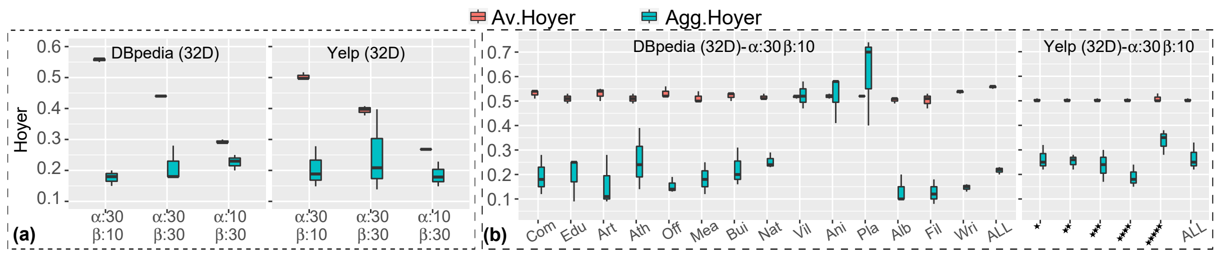
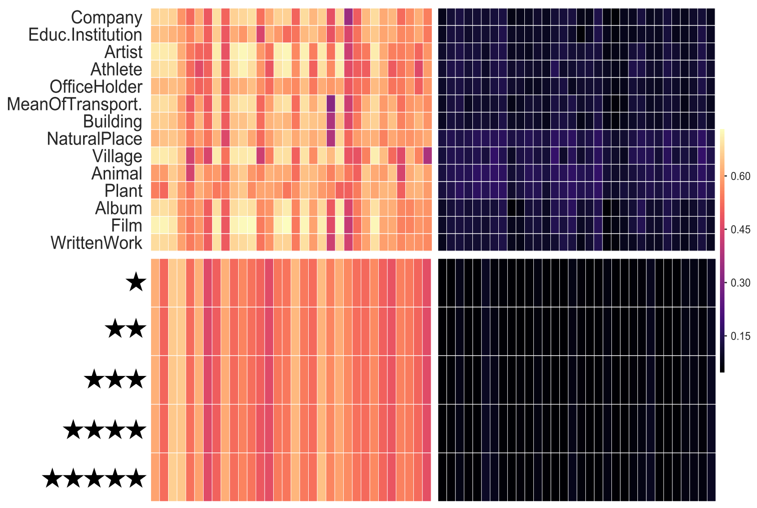
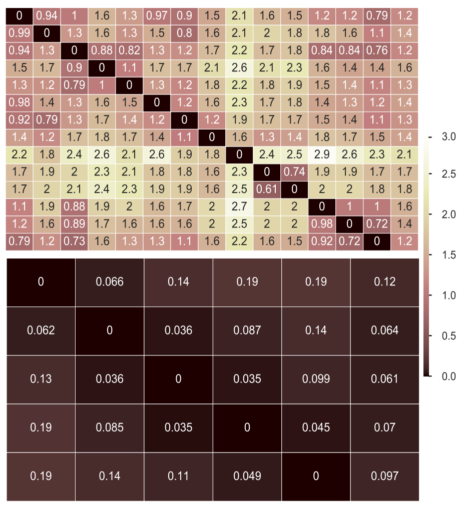


Figure 3: (a) Av.Hoyer of posterior samples vs. Hoyer of the aggregated posterior samples (Agg.Hoyer). Box plots are generated based on 3 runs. (b) Same quantities but plotted per class.



(a) Heat map for the average means of the Beta Distributions per class (Y-axis). Top: DBpedia, Bottom: Yelp, Left:(α=30, β=10), Right:(α=10, β=30)



(b) Heat map for the cross-class KL based on word distributions. KL values reported in cells. Last column is the average per class.

Figure 4: Left: heat maps of γ_{class} (§4.2.1) . Right: KL divergence between the word distributions of the classes.

γ_{class} are small) encouraging the model to propagate all the information about the sentences.

Analysis of z. To complement the results of the previous experiment, in Figure 3 (b) we report Av.Hoyer and Agg.Hoyer of the test sentences calculated for each class. Closeness of the two quantities, indicates the only the class information is encoded in sparsity pattern, while lower Agg.Hoyer compared to Av.Hoyer means the class information is likely to be mixed with other information.

For DBpedia the ratio of the Av.Hoyer and Agg.Hoyer varies. For Yelp, Agg.Hoyer is lower than the Av.Hoyer and remains approximately the same value for all classes. These findings accord well with Figure 4 (a) showing a uniform patterns of γ_{class} for Yelp and distributed patterns for DBpedia.

4.3 Downstream Task: Text Classification

We also compare the performance of the sparse representations with their dense counterparts from vanilla VAE on two text classification tasks. Comparing the accuracy of the classifiers on the same level of reconstruction loss, illustrates that the performance of the sparse representation is on par with its dense counterpart on DBpedia (rec-loss: 107, HSVAE-Acc:0.61, VAE-Acc:0.62), whereas for Yelp we observe a drop (rec-loss: 377, HSVAE-Acc:0.36, VAE-Acc:0.48). This is expected since the sparsity patterns of Figure 4 illustrates a better separation across classes for DBpedia, making it an easier setup for the classifier. Details on accuracy and reconstruction loss are reported in Appendix F.

5 Conclusion

We present a novel VAE model - Hierarchical Sparse Variational Autoencoder (HSVAE), capable of learning representations reflecting the underlying sparsity and distributedness characteristics of two text classification corpora. Empirically, we observed that for the sparsity patterns to encode the classes of the sentences, classes should be distinctive enough. This suggests that prior to use of the corpus its underlying statistical properties should be properly understood and curated. Additionally, a more systematic way of tuning the sensitivity of the model to capture different characteristics of a corpus is a potential research direction.

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, 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.

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.

Adam Coates and Andrew Y. Ng. 2011. The importance of encoding versus training with sparse coding and vector quantization. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, pages 921–928. Omnipress.

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.

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*.

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

Zachary Chase Lipton. 2016. The mythos of model interpretability. *CoRR*, abs/1606.03490.

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*.

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 Pratim Talukdar, and Tom M. Mitchell. 2012. Learning effective and interpretable semantic models using non-negative sparse embedding. In *COLING 2012, 24th International Conference on Computational Linguistics, Proceedings of the Conference: Technical Papers, 8-15 December 2012, Mumbai, India*, pages 1933–1950. Indian Institute of Technology Bombay.

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.

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

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.

Jakub M. Tomczak and Max Welling. 2018. Vae with a vampprior. In *Proceedings of the International Conference on Artificial Intelligence and Statistics, pp. 1214âĂŞ1223.*

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.

David P Vinson and Gabriella Vigliocco. 2008. Semantic feature production norms for a large set of objects and events. *Behavior Research Methods*, 40(1):183–190.

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.

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.

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*, NIPSâĂŹ15, page 649âĂŞ657, Cambridge, MA, USA. MIT Press.

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:

$$\mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x)) = \int_{z,\gamma} dz d\gamma\, q_\phi(z,\gamma|x) \log \frac{q_\phi(z,\gamma|x)}{p_\theta(z,\gamma|x)}, \tag{3}$$

after rearranging terms in equation 3 we can obtain:

$$\log p_\theta(x) - \mathbb{D}_{KL}(q_\phi(z,\gamma|x)||p_\theta(z,\gamma|x)) = \underbrace{\int_{z,\gamma} dz d\gamma\, q_\phi(z,\gamma|x) \log \frac{p_\theta(z,\gamma,x)}{q_\phi(z,\gamma|x)}}_{\text{ELBO}}, \tag{4}$$

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:

$$\int_{z,\gamma} dz d\gamma\, q_\phi(z|\gamma,x)q_\phi(\gamma|x) \log \frac{p_\theta(x|z)p_\theta(z|\gamma)p_\theta(\gamma)}{q_\phi(z|\gamma,x)q_\phi(\gamma|x)}, \tag{5}$$

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

$$\begin{gathered}\int_{z,\gamma} dz d\gamma\, q_\phi(z|x,\gamma)q_\phi(\gamma|x) \log p_\theta(x|z) \\ \int_\gamma d\gamma\, q_\phi(\gamma|x) \int_z dz\, q_\phi(z|x,\gamma) \log p_\theta(x|z) \therefore \\ \left\langle \int_z dz\, q_\phi(z|x,\gamma) \log p_\theta(x|z) \right\rangle_{q_\phi(\gamma|x)} \therefore\end{gathered} \tag{6}$$

The second term is:

$$\begin{gathered}\int_{z,\gamma} dz d\gamma\, q_\phi(z|x,\gamma)q_\phi(\gamma|x)[\log q_\phi(z|x,\gamma) - \log p_\theta(z|\gamma)] \\ \left\langle \int_z dz\, q_\phi(z|x,\gamma)[\log q_\phi(z|x,\gamma) - \log p_\theta(z|\gamma)] \right\rangle_{q_\phi(\gamma|x)} \therefore \\ \left\langle \mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma)) \right\rangle_{q_\phi(\gamma|x)} \therefore\end{gathered} \tag{7}$$

Finally, the third term is:

$$\int_{z,\gamma} dz d\gamma\; q_\phi(z|x,\gamma) q_\phi(\gamma|x)[\log q_\phi(\gamma|x) - \log p_\theta(\gamma)]$$
$$\int_{\gamma} d\gamma\; q_\phi(\gamma|x)[\log q_\theta(\gamma|x) - \log p_\theta(\gamma)]\times$$
$$\times \underbrace{\int_z dz\, q_\phi(z|x,\gamma)}_{\text{sums to 1 for each}:\gamma} \therefore$$
$$\int_\gamma d\gamma\; q_\phi(\gamma|x)[\log q_\phi(\gamma|x) - \log p_\theta(\gamma)]\therefore$$
$$\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma))\therefore \tag{8}$$

Collecting all the three terms into the single ELBO:

$$\left\langle \int_z dz\, q_\phi(z|x,\gamma) \log p_\theta(x|z) \right\rangle_{q_\phi(\gamma|x)} - \\ -\left\langle \mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma)) \right\rangle_{q_\phi(\gamma|x)} - \\ -\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)), \tag{9}$$

Using Monte Carlo approximation of the expectations:

$$\frac{1}{N}\sum_{\gamma\sim q_\phi(\gamma|x)}^{N}\left[\frac{1}{M}\sum_{z\sim q_\phi(z|x,\gamma)}^{M}\log p_\theta(x|z)\right] - \\ -\frac{\psi}{N}\sum_{\gamma\sim q_\phi(\gamma|x)}^{N}\left[\mathbb{D}_{KL}(q_\phi(z|x,\gamma)||p_\theta(z|\gamma))\right] - \\ -\lambda\mathbb{D}_{KL}(q_\phi(\gamma|x)||p_\theta(\gamma)), \tag{10}$$

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 $p_2(z)\sim\mathcal{N}(z;0,\sigma\rightarrow 0)$ 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:

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

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

D Sampling of z

In this section we briefly outline how we sample the latent variable z from the graphical model shown in Figure 1 (b).

D.1 Sampling: Training of the Model

for each x_i in a batch $\{x_1, ..., x_k\}$:

1. sample one γ_i from $q_\phi(\gamma|x_i)$
2. sample one z_i from $q_\phi(z|x_i,\gamma_i)$

The number of sampled γ_i and z_i is equal to the the number of x_i's in the batch.

D.2 Sampling: Evaluation of Hoyer

To evaluate Hoyer either on validation or test dataset for each x_i in the dataset $\{x_1, ..., x_n\}$ we first obtain its corresponding z_i (the procedure is described in Section D.1) such that $x_1 -> z_1$. Then we normalise $\bar{z}_i = z_i/\sigma(z)$, where $z =$

Models	**Yelp**		**DBPedia**		**Yelp**		**DBPedia**	
	R (16 D)	R (32 D)	R (16 D)	R (32 D)	Acc. (16 D)	Acc. (32 D)	Acc. (16 D)	Acc (32 D)
VAE$_{C=0}$	386.7 ± 2.4	386.0 ± 0.8	115.7 ± 0.5	115.7 ± 0.9	0.21 ± 0.02	0.21 ± 0.00	0.07 ± 0.00	0.07 ± 0.00
VAE$_{C=5}$	379.3 ± 0.5	380.0 ± 0.0	110.0 ± 0.8	110.7 ± 0.5	0.45 ± 0.02	0.46 ± 0.01	0.50 ± 0.03	0.44 ± 0.02
VAE$_{C=10}$	374.7 ± 0.5	377.3 ± 0.9	106.7 ± 0.5	107.7 ± 0.5	0.48 ± 0.00	0.47 ± 0.00	0.58 ± 0.01	0.62 ± 0.02
VAE$_{C=15}$	372.0 ± 0.0	373.7 ± 1.3	104.0 ± 0.8	105.7 ± 0.5	0.50 ± 0.00	0.50 ± 0.01	0.66 ± 0.02	0.66 ± 0.02
HSVAE$_{\alpha=30,\beta=10}$	378.0 ± 0.8	377.7 ± 0.9	108.3 ± 0.5	107.0 ± 0.0	0.36 ± 0.06	0.36 ± 0.01	0.59 ± 0.03	0.61 ± 0.00
HSVAE$_{\alpha=30,\beta=30}$	374.0 ± 0.0	371.7 ± 1.3	107.0 ± 0.0	104.7 ± 0.5	0.35 ± 0.01	0.46 ± 0.05	0.63 ± 0.03	0.72 ± 0.03
HSVAE$_{\alpha=10,\beta=30}$	365.0 ± 0.0	353.7 ± 0.5	98.3 ± 0.7	96.0 ± 2.2	0.49 ± 0.01	0.54 ± 0.00	0.81 ± 0.02	0.85 ± 0.01
SC	-	-	-	-	0.53 ± 0.00	0.54 ± 0.00	0.98 ± 0.00	0.97 ± 0.00

Table 1: Table shows the reconstruction (R) and the classification results for the vanilla VAE and HSVAE. To report the classification results we use accuracy (Acc.). We also experiment with the number of dimensions in the latent code of the VAEs: 16 D and 32 D. SC stands for a simple classifier.

$\{z_1, ..., z_n\}$. Finally, for each $\bar{z}_i$ we compute Hoyer such that $Hoyer(\bar{z}_i) = \frac{\sqrt{d} - ||\bar{z}_i||_1 / ||\bar{z}_i||_2}{\sqrt{d}-1}$, where d is the dimensionality of $\bar{z}_i$. To report the Hoyer for the whole dataset we compute Av.Hoyer $= \frac{1}{N}\sum_i^N Hoyer(\bar{z}_i) = \frac{\sqrt{d} - \frac{1}{N}\sum_i^N ||\bar{z}_i||_1 / ||\bar{z}_i||_2}{\sqrt{d}-1}$.

D.3 Sampling: Classification Experiment

for each x_i in a batch $\{x_1, ..., x_k\}$:

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

D.4 Sampling: Ratio Experiment

Main Text: Figure 3 (a) Numerator is calculated in the same way as in Section D.2. To calculate the Hoyer for the denominator we first average over all the $\{\bar{z}_1, ..., \bar{z}_n\}$ that we used in the numerator i.e. $m = \frac{1}{N}\sum_{i=1}^N \bar{z}_i$ and then measure the Hoyer on m. Therefore the ratio is: $\frac{1}{N}\sum_i^N Hoyer(\bar{z}_i)/Hoyer(m)$

Main Text: Figure 3 (b) First we compute the following for the test corpus:

1. compute b_i = Binarize($|\bar{z}_i|$) (we use the threshold of 1×10^{-2} to return 1 if the dimension of $|\bar{z}_i|$ larger than the threshold and 0 otherwise.)
2. compute the mean of b_i: $\bar{b} = \frac{1}{N} b_i$
3. define the threshold t as e.g. the 5% percentile of the entries in $\bar{b} \in \mathbb{R}^D$
4. threshold $\bar{b}$: $\tilde{b}_d = \delta(\bar{b}_d > t), \tilde{b} = (\tilde{b}_1, ..., \tilde{b}_D)$
5. compute $Hoyer(m \odot \tilde{b})$ as the denominator

Then, for the test sentences of each class we repeat the procedure, however this time $\bar{b}$ in the step 4. is being reused for the all classes. In other words the $\bar{b}$ is the same for whole test corpus and for the test sentences of each class.

E How we obtain γ_{class}

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 for each class we average its $\mu_{\gamma(x)}$ vectors to obtain a single vector that represent this class: $\gamma_{class} = \frac{1}{M}\sum_{x \in class} \mu_{\gamma(x)}$, where M is a number of sentences in the class.

F Text Classification

We compare performance of the sparse z with it dense counterpart (vanilla VAE) on the text classification task (Table 1). Reconstruction error of VAEs and the accuracy are highly correlated. Therefore, we find it necessarily to compare z of VAEs that have similar reconstruction. Corpus and a number of latent dimensions are also among the factors that play a role. For DBPedia, z for 32 dimensions is only slightly worse than the dense z, while for 16 dimensions sparse z outperforms it. But, this is not the case for Yelp. We partially attribute it to the properties of corpus that γ captures.

G Hoyer

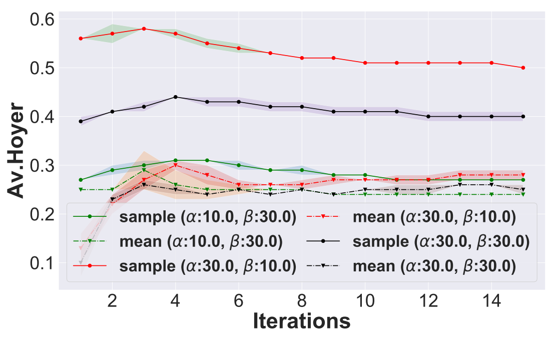


Figure 5: Average Hoyer (Av.Hoyer) on Yelp corpus dev set for HSVAE. Lines are an average over the 3 runs of the models, the shaded area is the standard deviation.

H Reproducibility Checklist

H.1 Hardware

Please refer to Table 2 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 2: Computing infrastructure.

H.2 Datasets

In our experiments we use two corpora Yelp and DBpedia. We download Yelp[4](Yang et al., 2017) as it is and do not perform any additional preprocessing. As for DBpedia (Zhang et al., 2015) we first download[5] the corpus and then perform the preprocessing. The preprocessing is following: 1) remove all non-ASCII characters, 2) remove all quotations marks, 3) remove all hyperlinks, 4) perform tokenisation with spaCy[6] and 5) convert all the tokens to lower case. Then, for each class we randomly sample 10,000 sentences for the training corpus and 1,000 sentences for the test and validation respectively. We also reduce the vocabulary size to the first 20,000 most frequent words.

[4]The link to download the corpus https://github.com/jxhe/vae-lagging-encoder/blob/master/prepare_data.py.

[5]The link to download the corpus https://github.com/srhrshr/torchDatasets/blob/master/dbpedia_csv.tar.gz

[6]https://spacy.io

	Yelp	DBpedia
# sent. (train corpus)	100,000	140,000
# sent. (valid corpus)	10,000	14,000
# sent. (test corpus)	10,000	14,000
vocabulary size	19,997	20,000
min sent. length.	20	1
av. sent. length.	96	35
max. sent. length.	200	60
# classes	5	14
# sent. in each class	2000	1000

Table 3: Statistics of corpora. Vocabulary size excludes the ⟨pad ⟩and ⟨EOS ⟩symbols.

H.3 Model

For both encoder and decoder we use the GRU (Cho et al., 2014) network. We couple the encoder with the decoder by concatenating z with the word embedding at each time step. The word embeddings are the same for encoding and decoding. To train the model we use the Adam optimiser (Kingma and Ba, 2014). To sample from the posterior distribution $q(z|x, \gamma)$ we use Binary Concrete distribution (Maddison et al., 2016). For the hyper-parameters that we use please refer to Table 4.

	Yelp	DBpedia
learning rate	8×10^{-4}	8×10^{-4}
batch size	256	512
hidden dim. γ	32	32
hidden dim. z	32	32
encoder dim.	512	512
decoder dim.	512	512
word embeddings dim.	256	256
# epochs	15	15
ψ	0.01	0.01
λ	0.01	0.01
temperature of Binary Concrete	0.5	0.5
# parameters	17,859,487	17,861,794
time per epoch (sec.)	352	165

Table 4: Hyper-parameters we use to train the models.

To select the hyper-parameters we use a manual tuning. Batch size: we simply select the largest batch size that can fit into the memory of the GPU. Encoder, decoder and hidden dims: we roughly follow (Yang et al., 2017), however we halve the number of parameters of encoder and decoder in order to be able to fit the model into the GPU. Learning rate and number of epochs: we use the vanilla VAE with the collapsed KL term (we simulated the procedure by setting $C = 0$ in the objective function of Burgess et al. (2018)) to decide on the learning rate and the number of epochs. It is well known that when the KL term of the vanilla VAE is zero the decoder do not use the information provided by the encoder. Thus, it serves as good baseline to understand if the decoder of a VAE that one trains uses the information of the encoder or not. We select the learning rate in tandem with the number of epochs. With the chosen, aforementioned, parameters the vanilla VAE has enough training iterations before it starts overfitting on the validation data. For us to be able to compare with this baseline we use the same parameters in HSVAE.