LangVAE and LangSpace: Building and Probing for Language Model VAEs

Danilo S. Carvalho[1], **Yingji Zhang**[2], **Harriet Unsworth**[1], **André Freitas**[1,2,3]
National Biomarker Centre, CRUK-MI, University of Manchester, United Kingdom[1]
Department of Computer Science, University of Manchester, United Kingdom[2]
Idiap Research Institute, Switzerland[3]
https://github.com/neuro-symbolic-ai/{LangVAE, LangSpace} | Short Video

Abstract

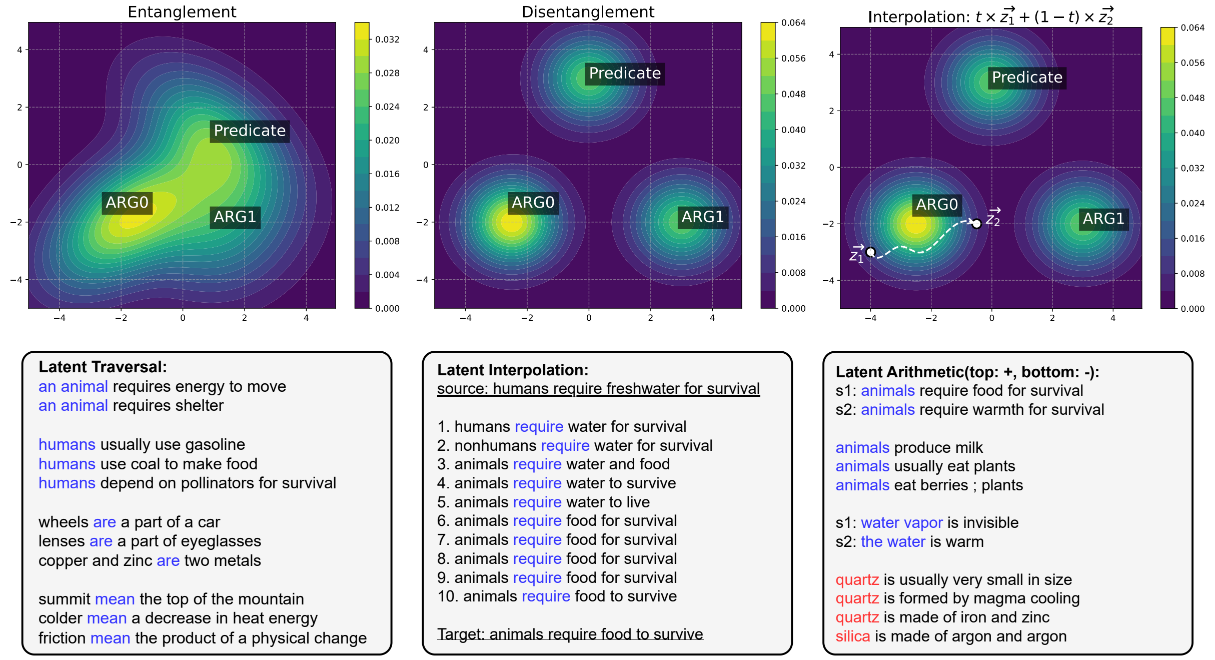
We present *LangVAE*, a novel framework for modular construction of variational autoencoders (VAEs) on top of pre-trained large language models (LLMs). Such language model VAEs can encode the knowledge of their pre-trained components into more compact and semantically disentangled representations. The representations obtained in this way can be analysed with the LangVAE companion framework: *LangSpace*, which implements a collection of probing methods, such as vector traversal and interpolation, disentanglement measures, and cluster visualisations. LangVAE and LangSpace offer a flexible, efficient and scalable way of building and analysing textual representations, with simple integration for models available on the HuggingFace Hub. Additionally, we conducted a set of experiments with different encoder and decoder combinations, as well as annotated inputs, revealing a wide range of interactions across architectural families and sizes w.r.t. generalisation and disentanglement. Our findings demonstrate a promising framework for systematising the experimentation and understanding of textual representations.

1 Motivation and Purpose

Variational Autoencoders (VAEs) (Kingma et al., 2013) are of considerable importance in machine learning due to their capacity to integrate prior knowledge, quantify uncertainty, enhance generalisation, and deliver interpretability. First, the integration of prior distribution serves as an inductive bias, enabling the model to leverage existing knowledge and providing a principled way to incorporate domain expertise. In the computational linguistics domain, for example, the hierarchical syntax information can be well encoded via hyperbolic prior (Davidson et al., 2018; Cho et al., 2023). Second, their probabilistic formulation allows for explicit uncertainty quantification, providing not only point estimates but also confidence intervals over latent variables and reconstructions, which is significant in the Safety and Trustworthy AI domain, such as hallucinations of LLMs (Ji et al., 2023). Third, by enforcing a smooth and continuous latent space, VAEs promote better composition and generalisation, as they capture the underlying generative factors of the input distribution (Bonnet and Macfarlane, 2024). Fourth, the latent space can compress the knowledge into abstract-level concepts, which is similar to how humans understand the world (Barrault et al., 2024). Concurrently, the rapid pace of development of LLMs has led to substantial gains in a wide variety of NLP tasks, demonstrating remarkable knowledge representation capabilities (Kauf et al., 2023; Selby et al., 2025), but present critical challenges in interpretability and fine-grained control (Kunz and Kuhlmann, 2022; Friedman et al., 2024).

To leverage the strengths of both LMs and VAEs, Language model-based VAEs (LM-VAEs) (Bowman et al., 2015) have been proposed and widely deployed in the controlled text generation domain, such as style transfer tasks: modifying sentences with regard to markers of sentiment, formality, affirmation/negation (Bao et al., 2019; Vasilakes et al., 2022; Gu et al., 2022; Liu et al., 2023; Gu et al., 2023; Liu et al., 2024) and textual, syntactic, semantic representation learning domain (Mercatali and Freitas, 2021; Carvalho et al., 2023; Zhang et al., 2024b,c). However, despite their strategic positioning in delivering more controlled latent representations, there has been limited software infrastructure support to facilitate experimentation with LM-VAEs and in particular, scaling-up to Large Language Model configurations (LLM-VAEs).

In this work we address these issues by presenting a novel framework for modular construction of LM-VAEs on top of pre-trained LMs of different scales, called *LangVAE*, and its companion framework *LangSpace*, dedicated to latent space probing and evaluation. LangVAE introduces a novel ap-



Latent Traversal:
an animal requires energy to move
an animal requires shelter

humans usually use gasoline
humans use coal to make food
humans depend on pollinators for survival

wheels are a part of a car
lenses are a part of eyeglasses
copper and zinc are two metals

summit mean the top of the mountain
colder mean a decrease in heat energy
friction mean the product of a physical change

Latent Interpolation:
source: humans require freshwater for survival

1. humans require water for survival
2. nonhumans require water for survival
3. animals require water and food
4. animals require water to survive
5. animals require water to live
6. animals require food for survival
7. animals require food for survival
8. animals require food for survival
9. animals require food for survival
10. animals require food to survive

Target: animals require food to survive

Latent Arithmetic(top: +, bottom: -):
s1: animals require food for survival
s2: animals require warmth for survival

animals produce milk
animals usually eat plants
animals eat berries ; plants

s1: water vapor is invisible
s2: the water is warm

quartz is usually very small in size
quartz is formed by magma cooling
quartz is made of iron and zinc
silica is made of argon and argon

Figure 1: *LangVAE* is a flexible framework designed to support arbitrary combinations of pretrained encoders and decoders for learning latent representations under either a categorical semantic prior or a Gaussian prior. *LangSpace* facilitates comprehensive analysis of the learned latent space through automated evaluation of key properties such as disentanglement and visualization (top) and enables controlled generation by leveraging these latent properties, such as latent traversal, interpolation, and arithmetic operations (bottom).

proach for latent vector unpooling to autoregressive LMs that sharply reduces the computational and memory requirements, while incorporating compatibility to contemporary LLMs and hardware optimisations.

Finally, we conducted a set of experiments as a case study to demonstrate the frameworks' capabilities and highlight the effects of different combinations of encoder and decoder models, in terms of generalisation and latent space disentanglement, evidencing the impact of facilitating a systematic analysis across different encoder-bottleneck-decoder combinations and parametrisations.

Both frameworks are available as python libraries in the PyPI package repository and on public source code repositories[1] [2]. A demonstration video is available at: youtu.be/DVcrdIX9CfI.

2 Language Model VAEs

A language model VAE (LM-VAE) is a variational autoencoder where both the encoder and decoder components are LMs (Bowman et al., 2015; Li et al., 2020; Tu et al., 2022; Zhang et al., 2023). It can encode the knowledge of their pre-trained components into compact latent vectors and enables guided language generation from an abstract level using said vectors. The benefits of such models also extend to interpretability (due to their better disentanglement properties), as the VAE architectural bottleneck provides a singular point for probing a model's latent space structure and its syntactic/semantic representation (Li et al., 2020; Mercatali and Freitas, 2021; Carvalho et al., 2023; Zhang et al., 2024b,c) and inferential properties (Bonnet and Macfarlane, 2024). The creation of continuous latent representation spaces, with better disentanglement and separability of syntactic/semantic properties offers a key mechanism for supporting generative control both at the level of sentences (Bao et al., 2019; Felhi et al., 2022; Zhang et al., 2024c) and natural language inferences (Yu et al., 2022).

In its most basic conceptualisation, an LM-VAE consists of: (**a**) an encoder type LLM (e.g., BERT, T5), to provide base representations for each token of the input text; (**b**) a pooling process to accumulate the input token representations; (**c**) a projection layer, to convert the base encoding to the regularised VAE latent space; (**d**) an unpooling process, to derive token representations from a latent vector

[1] https://github.com/neuro-symbolic-ai/LangVAE/
[2] https://github.com/neuro-symbolic-ai/LangSpace

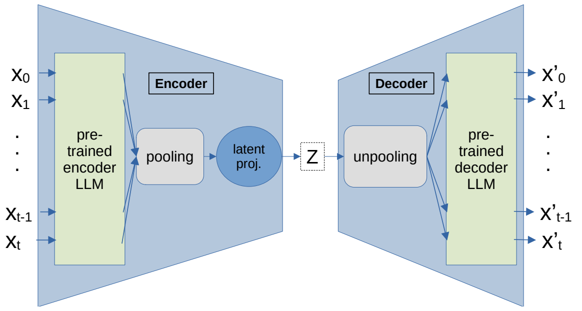


Figure 2: Diagram of fundamental LLVAE architecture.

and feed them to the decoder; and (**e**) a decoder type LLM (e.g., GPT, Llama) capable of generating tokens from a sequence of input representations. This structure is illustrated in Figure 2. On the top of this base configuration, syntactic and semantic features can be injected into the latent space, aiming to improve the localisation and control of such features via conditionalisation mechanisms, such as CVAE or clustering losses. Moreover, further architectural interventions can be integrated aiming for additional control, such as the addition of INN layers (Zhang et al., 2024a), aiming for improving the separability of semantic features.

2.1 Optimus

The pioneer LLVAE is Optimus (Li et al., 2020), which combines a BERT encoder and a GPT-2 decoder to perform sentence encoding, using a mean pooling process, a linear projection layer (MLP), and a unpooling process consisting of two concurrent schemes for latent memory injection to the decoder:

Memory: appends a projection of the latent vector directly to each hidden layer of the decoder as a hidden memory vector for the decoder to attend.

Embedding: adds a projection of the latent vector to the decoder embedding layer at each decoding step.

Optimus is trained end-to-end, meaning that the encoder projection layer and the memory and embedding injection layers are jointly trained with the base encoder and decoder models. In this way, the pre-trained models are fine-tuned to "weld" with the projection and injection layers, facilitating convergence.

Despite its demonstrated capabilities and potential, Optimus has some limitations, in particular regarding model coupling and scalability. We next discuss our proposed approach for building LM-VAEs and its improvements over the SOTA.

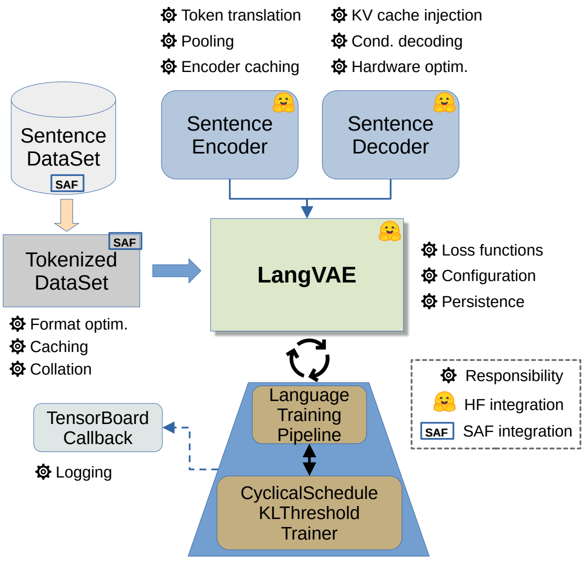


Figure 3: Overview of the LangVAE framework.

3 LangVAE: Building modular LM-VAEs

Aiming to address current LM-VAE limitations and facilitate the development of specialised models and experimentation over next-gen LLMs, we developed *LangVAE*. This is a novel framework specifically targeted at LM-VAE research, focused on the modular development of the architectural components discussed in the previous section (especially projections and unpooling processes), and having a strong integration with the python transformers library[3]. LangVAE is developed and distributed as a python library[1] under the GPLv3 License. It is built on top of the pythae library for autoencoders (Chadebec et al., 2022). Figure 3 provides an overview of LangVAE's modules and responsibilities.

3.1 Architecture

LangVAE implements the fundamental LM-VAE architecture (Figure 2) in the following ways:

Pre-trained LLM encoder: as a loader for an encoder type LLM compatible with the transformers library, via the automodel classes (AutoModel, AutoModelForTextEncoding).

Pooling process: mean pooling, last hidden state of the base encoder, or the CLS token hidden state, which is automatically selected depending on the pre-trained encoder model configuration.

Latent projection layer: a linear MLP that adjusts the input encoding size on training time.

[3] https://github.com/huggingface/transformers

Unpooling process: a variation of the memory injection scheme from Optimus, called *KV cache injection*, which does not require customisation of the pre-trained decoder code. Instead, it uses the transformers library KV caching mechanism for guiding the decoder (detailed in the next section).

Pre-trained LLM decoder: same as the encoder, but relying on the transformers *AutoModelForCausalLM* class for model parametrisation regarding tokenizer configuration and hardware optimisations (e.g., flash attention and multi-GPU distribution).

In addition, LangVAE provides the following functionalities:

Data conversion: TokenizedDataSet classes for convenient and efficient tokenisation of text datasets, including the handling of annotations.

Training pipeline: supporting cyclical schedule KL annealing to avoid the KL vanishing problem, with beta and KL thresholds.

Training monitoring: with tensorboard logging.

3.2 KV cache injection

One of the central contributions behind LangVAE is the key-value (KV) cache injection scheme, as an alternative to Optimus' memory injection. This new scheme uses the KV caching mechanism of the Causal LM model classes within the transformers library to inject a positional projection of the latent vector. A linear projection of the latent vector $h_{cache} = W_m z$ plays the role of an additional context to guide generation, in the form of hidden KV cache entries X_t^h interleaved with those produced by the decoder, where $W_m \in \mathbb{R}^{LH \times S}$ is separated into $S \times L$ (sequence length * # layers) vectors of hidden size $H = K \times V$. Figure 4 illustrates this scheme.

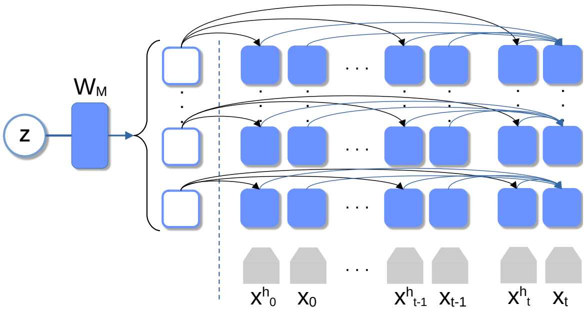


Figure 4: Illustration of the KV cache injection scheme. $W_m z$ projects hidden KV cache entries X_t^h that are attended by the decoder when predicting the next token. The hidden cache entries are interleaved with the ones produced by the decoder.

There are two main advantages to this approach. Firstly, it eliminates the need to change the layout of the hidden layers to accommodate the injected memory vector. Therefore, it is compatible with any model that supports the KV caching mechanism. Lastly, in enables training the LM-VAE model with the weights for the base pretrained models frozen, greatly reducing the computational and memory requirements. Additionally, this scheme allows distributed training of the injection layer, as the projection matrices can be co-located with the respective hidden layers in training time, and size of context (number of hidden cache entries) can be adjusted.

3.3 Main advantages & Limitations

The main advantages of LangVAE can be summarised as follows:

- Modular architecture allows flexible development of different LM-VAE configurations. Flexible composition of base models and bottleneck parametrisations, loss functions, etc.
- Compatible with most state-of-the-art autoregressive models.
- Has a substantially reduced computational requirements for training, compared to the SOTA LM-VAE (Optimus), with an average parameter reduction of over 95% measured when using decoder models between 3B to 7B parameters (Section 5.1).
- Supports multi-GPU training and inference.

Its main limitations are related to the cache injection mechanism:

- Slower convergence, as there are far less parameters to adjust.
- Latent vector sizes tend to be larger, compared to Optimus, to compensate for the overall parameter reduction.

3.4 Installation and API Examples

LangVAE can be installed directly from the PyPI package repository with: $pip\ install\ langvae$

We briefly illustrate the key components of LangVAE's API and how they are instantiated in the supplementary material (Appendix Section A.1). A full example of model training can be found in the README file of the code repository[1] and on the supporting python notebook[4].

4 LangSpace: Simplified probing for LM-VAEs

LangSpace[2], is a companion framework to LangVAE focused on the evaluation and on the latent space probing for LM-VAEs. It provides an easy-to-use API to perform a variety of analyses on pre-trained LM-VAEs models, namely:

- Probes: vector arithmetic and interpolation, latent space traversal, disentanglement and cluster visualisation.
- Metrics: disentanglement (z-diff, z-min-var, MIG, Disentanglement, Informativeness, Completeness), interpolation (quality, smoothness).

4.1 Installation and API Examples

Like LangVAE, LangSpace can be installed from the PyPI repository with: *pip install langspace*

We briefly illustrate below the use of one of LangSpace probes: latent traversal. A full example with all the available probes can be found within this public notebook[5]

```
# Seed sentences to traverse
sentences = [
    "animals require food to survive",
    "water vapor is invisible"
]
# Dataset importer
ds = ListImporter()(
    [sent.split() for sent in sentences]
).sentences
# Create dataset tokeniser
seeds = TokenizedDataSet(
    ds, model.decoder.tokenizer,
    model.decoder.max_len
)
# Create probe and generate report: a
# dataframe with collumns for the
# original sentence, traversed dimension,
# distance traversed and the generated
# result, respectively.
trav_report = TraversalProbe(
    model, trav_dataset,
    sample_size=10,
    dims=list(range(128))
).report()
```

5 Case study & Model availability

To demonstrate LangVAE and LangSpace capabilities and highlight the effects of different combinations of encoder and decoder models, in terms of generalisation and disentanglement of the latent space, we conducted a set of experiments as a case study. The experiments consist of a simple explanation sentence modeling task (Zad et al., 2021; Dalvi et al., 2021) with posterior evaluation of the induced latent space. Pre-trained checkpoints for all model combinations presented in this study are available in our public HF Hub repository[6].

5.1 Experimental setup

For the pre-trained LLMs, we selected three distinct encoder models, in order of parameter size: BERT (base-cased) (Devlin et al., 2019), Flan-T5 (base) (Chung et al., 2024) and Stella (en-1.5B_v5) (Zhang et al., 2025), and four decoder models: GPT-2 (base) (Radford et al.), Qwen (2.5-3B) (Team, 2024), Llama (3.2-3B) (Grattafiori et al., 2024) and Mistral (7B-v0.3) (Jiang et al., 2023). The selection considered the inclusion of different model families and sizes. For each combination, inputs without and with semantic role labeling (SRL) annotations were used (as semantic features), where the SRL annotations were passed as additional variables (one-hot encoded) to the encoder only, going through a separate pooling process (always mean pooling). The latent size (128) and maximum sentence was kept the same for all tests. All models were trained for 50 epochs, with $LR = 0.001$, $target\_kl = 2.0$, $max\_beta = 1.0$, 40 beta annealing cycles, and batch size of 50. Disentanglement measurements were obtained using LangSpace's disentanglement probe for the metrics z-diff (Higgins et al., 2017), z-min-var (Kim and Mnih, 2018) and Informativeness (Eastwood and Williams, 2018).

All experiments were performed on a computer with the following specifications: CPU: AMD EPYC 7413 24-Core, GPU: 2x NVIDIA A100-SXM4-80GB, Memory: 200GB. LangVAE allows caching of the base encoder outputs, which causes the training time to be mostly dominated by the base decoder inference time. The shortest training time was of aprox. 1h (GPT-2) and the longest was about 4.5h (Mistral-7B). Training requirements for larger decoders scale similarly to inference, with a training run of Phi-4 (14B) also taking about 4.5h

[4] https://bit.ly/3FMPg5N

[5] https://bit.ly/424bjw3

[6] https://huggingface.co/neuro-symbolic-ai

to complete. The ratios of the LangVAE trained models' size to the base LLMs was: GPT-2 = 0.547, Qwen2.5-3B = 0.024, Llama3.2-3B = 0.076, Mistral-7B = 0.037. Excluding GPT-2, this represents an over 95% parameter reduction.

5.2 Data

The same data was used for all tests: a subset of all explanatory sentences from the EntailmentBank dataset (Dalvi et al., 2021), which was loaded using the *saf-datasets*[7] library. The dataset contains 12496 sentences, from which 99% were used for training and 1% for validation[8]. Evaluation was performed on a random sample of 200 sentences including the validation set and a small portion of the training set.

SRL annotation was performed using the AllenNLP[9] library with a SOTA SRL model (Shi and Lin, 2019).

5.3 Results

The results for the explanation sentence modeling task are presented in Table 1. The first observation is that the highest reconstruction performance was achieved by the smallest model combination (for SRL). While not the expected outcome, this can be explained by the constraint imposed on the latent space size, composed with the limited training data, causing the simpler model to better generalise the inputs.

The encoder complexity has a substantial impact on the generalisation capability of the model: even though bert-base-cased and flan-t5-base have the same encoding size (768), BERT outperforms T5 in most cases, indicating a higher level of information entanglement on T5. Stella, on the other hand, has a much larger encoding size (1536), with a larger dominating effect based on the information loss over the dimensionality reduction.

The injection of the SRL categories within the model improved reconstruction performance in all combinations except when Mistral is the decoder. This is a surprising result and indicate some particularity of Mistral's internal representations that invite further investigation.

Finally, the SRL categories did not induce consistent improvements on the disentanglement scores, with the exception of Llama3.2, where it led to qualitative improvements, as illustrated in Figure 5 (Appendix B).

Encoder	**Decoder**	**Annot.**	**Reconstr.**	**Disentanglement**		
			(BLEU)	z-diff	z-m-var ↓	inform.
BERT	gpt-2	-	0.76	0.46	0.68	0.36
BERT	gpt-2	SRL	**0.84**	0.43	0.70	0.40
BERT	Qwen	-	0.44	0.58	0.69	0.46
BERT	Qwen	SRL	0.49	0.53	0.61	0.44
BERT	Llama	-	0.65	**0.62**	0.71	0.38
BERT	Llama	SRL	0.80	0.59	0.65	0.43
BERT	Mistral	-	0.81	0.51	**0.59**	0.43
BERT	Mistral	SRL	0.75	0.55	0.62	0.44
Flan-T5	gpt-2	-	0.11	0.50	0.62	0.35
Flan-T5	gpt-2	SRL	0.81	**0.62**	0.67	0.42
Flan-T5	Qwen	-	0.19	0.52	0.69	0.39
Flan-T5	Qwen	SRL	0.31	0.55	0.68	0.43
Flan-T5	Llama	-	0.74	0.52	0.68	**0.49**
Flan-T5	Llama	SRL	0.80	0.59	0.64	0.41
Flan-T5	Mistral	-	0.78	**0.62**	0.61	0.39
Flan-T5	Mistral	SRL	0.72	0.51	0.63	0.43
Stella	gpt-2	-	0.18	0.50	0.68	0.34
Stella	gpt-2	SRL	0.61	0.52	0.65	0.40
Stella	Qwen	-	0.15	0.48	0.69	0.44
Stella	Qwen	SRL	0.27	0.54	0.66	0.43
Stella	Llama	-	0.45	0.51	0.73	0.40
Stella	Llama	SRL	0.64	**0.62**	0.72	0.42
Stella	Mistral	-	0.57	0.54	0.72	0.46
Stella	Mistral	SRL	0.55	0.51	0.71	0.39

Table 1: Results from the explanation sentence modeling experiments. Best values for each column in bold.

6 Conclusion

In this work we presented *LangVAE*, a modular and efficient library for building language model VAEs (LM-VAEs), and its companion framework *LangSpace*, dedicated to LM-VAE latent space control, probing and evaluation. With the goal of lowering the experimental barriers in this research area, it introduces a novel approach for latent vector unpooling to autoregressive LMs that sharply reduces the computational and memory requirements for training such models, along with a flexible code architecture which is oriented towards modern LLM development.

We demonstrated the capabilities of LangVAE and LangSpace with a set of experiments using different encoder and decoder combinations, as well as annotated inputs, which reveal a wide range of interactions across architectural families and sizes w.r.t. generalisation and disentanglement. Such interactions point to uncovered factors regarding the models' internal representation properties and how they exchange information.

[7] https://github.com/neuro-symbolic-ai/saf_datasets

[8] The validation split here is just a means to track the training progress for overfitting, since the dataset is small.

[9] https://github.com/allenai/allennlp-models

References

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xinyu Dai, and Jiajun Chen. 2019. Generating sentences from disentangled syntactic and semantic spaces. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6008–6019.

Loïc Barrault, Paul-Ambroise Duquenne, Maha Elbayad, Artyom Kozhevnikov, Belen Alastruey, Pierre Andrews, Mariano Coria, Guillaume Couairon, Marta R Costa-jussà, David Dale, et al. 2024. Large concept models: Language modeling in a sentence representation space. *arXiv preprint arXiv:2412.08821*.

Clément Bonnet and Matthew V Macfarlane. 2024. Searching latent program spaces. *arXiv preprint arXiv:2411.08706*.

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

Danilo Silva de Carvalho, Giangiacomo Mercatali, Yingji Zhang, and André Freitas. 2023. Learning disentangled representations for natural language definitions. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1371–1384.

Clément Chadebec, Louis Vincent, and Stephanie Allassonniere. 2022. Pythae: Unifying generative autoencoders in python - a benchmarking use case. In *Advances in Neural Information Processing Systems*, volume 35, pages 21575–21589. Curran Associates, Inc.

Seunghyuk Cho, Juyong Lee, and Dongwoo Kim. 2023. Hyperbolic vae via latent gaussian distributions. *Advances in Neural Information Processing Systems*, 36:569–588.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2024. Scaling instruction-finetuned language models. *Journal of Machine Learning Research*, 25(70):1–53.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Tim R Davidson, Luca Falorsi, Nicola De Cao, Thomas Kipf, and Jakub M Tomczak. 2018. Hyperspherical variational auto-encoders. *arXiv preprint arXiv:1804.00891*.

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.

Cian Eastwood and Christopher KI Williams. 2018. A framework for the quantitative evaluation of disentangled representations. In *6th International Conference on Learning Representations*.

Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. 2022. Towards unsupervised content disentanglement in sentence representations via syntactic roles. *arXiv preprint arXiv:2206.11184*.

Dan Friedman, Andrew Lampinen, Lucas Dixon, Danqi Chen, and Asma Ghandeharioun. 2024. Interpretability illusions in the generalization of simplified models. In *Proceedings of the 41st International Conference on Machine Learning*, pages 14035–14059.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyuan Zhang, Heng Gong, and Bing Qin. 2022. A distributional lens for multi-aspect controllable text generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 1023–1043, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yuxuan Gu, Xiaocheng Feng, Sicheng Ma, Lingyuan Zhang, Heng Gong, Weihong Zhong, and Bing Qin. 2023. Controllable text generation via probability density estimation in the latent space. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12590–12616, Toronto, Canada. Association for Computational Linguistics.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *ICLR*.

Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. 2023. Survey of hallucination in natural language generation. *ACM computing surveys*, 55(12):1–38.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023. Mistral 7b.

Carina Kauf, Anna A Ivanova, Giulia Rambelli, Emmanuele Chersoni, Jingyuan Selena She, Zawad Chowdhury, Evelina Fedorenko, and Alessandro Lenci. 2023. Event knowledge in large language models: the gap between the impossible and the unlikely. *Cognitive Science*, 47(11):e13386.

Hyunjik Kim and Andriy Mnih. 2018. Disentangling by factorising. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 2649–2658. PMLR.

Diederik P Kingma, Max Welling, et al. 2013. Auto-encoding variational bayes.

Jenny Kunz and Marco Kuhlmann. 2022. Where does linguistic information emerge in neural language models? measuring gains and contributions across layers. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4664–4676.

Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4678–4699, Online. Association for Computational Linguistics.

Guangyi Liu, Zeyu Feng, Yuan Gao, Zichao Yang, Xiaodan Liang, Junwei Bao, Xiaodong He, Shuguang Cui, Zhen Li, and Zhiting Hu. 2023. Composable text controls in latent space with ODEs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 16543–16570, Singapore. Association for Computational Linguistics.

Yi Liu, Xiangyu Liu, Xiangrong Zhu, and Wei Hu. 2024. Multi-aspect controllable text generation with disentangled counterfactual augmentation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9231–9253, Bangkok, Thailand. Association for Computational Linguistics.

Giangiacomo Mercatali and André Freitas. 2021. Disentangling generative factors in natural language with discrete variational autoencoders. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3547–3556, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners.

David Selby, Yuichiro Iwashita, Kai Spriestersbach, Mohammad Saad, Dennis Bappert, Archana Warrier, Sumantrak Mukherjee, Koichi Kise, and Sebastian Vollmer. 2025. Had enough of experts? quantitative knowledge retrieval from large language models. *Stat*, 14(2):e70054.

Peng Shi and Jimmy Lin. 2019. Simple bert models for relation extraction and semantic role labeling. *ArXiv*, abs/1904.05255.

Qwen Team. 2024. Qwen2.5: A party of foundation models.

Haoqin Tu, Zhongliang Yang, Jinshuai Yang, and Yongfeng Huang. 2022. Adavae: Exploring adaptive gpt-2s in variational auto-encoders for language modeling. *arXiv preprint arXiv:2205.05862*.

Jake Vasilakes, Chrysoula Zerva, Makoto Miwa, and Sophia Ananiadou. 2022. Learning disentangled representations of negation and uncertainty. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8380–8397, Dublin, Ireland. Association for Computational Linguistics.

Jialin Yu, Alexandra I Cristea, Anoushka Harit, Zhongtian Sun, Olanrewaju Tahir Aduragba, Lei Shi, and Noura Al Moubayed. 2022. Interaction: a generative xai framework for natural language inference explanations. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE.

Samira Zad, Maryam Heidari, Parisa Hajibabaee, and Masoud Malekzadeh. 2021. A survey of deep learning methods on semantic similarity and sentence modeling. In *2021 IEEE 12th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON)*, pages 0466–0472. IEEE.

Dun Zhang, Jiacheng Li, Ziyang Zeng, and Fulong Wang. 2025. Jasper and stella: distillation of sota embedding models.

Yingji Zhang, Danilo Carvalho, and André Freitas. 2024a. Learning disentangled semantic spaces of explanations via invertible neural networks. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2113–2134.

Yingji Zhang, Danilo Carvalho, Marco Valentino, Ian Pratt-Hartmann, and André Freitas. 2024b. Improving semantic control in discrete latent spaces with transformer quantized variational autoencoders. In *Findings of the Association for Computational Linguistics: EACL 2024*, pages 1434–1450.

Yingji Zhang, Danilo S Carvalho, Ian Pratt-Hartmann, and André Freitas. 2023. Llamavae: Guiding large language model generation via continuous latent sentence spaces. *arXiv preprint arXiv:2312.13208*.

Yingji Zhang, Marco Valentino, Danilo Carvalho, Ian Pratt-Hartmann, and André Freitas. 2024c. Graph-induced syntactic-semantic spaces in transformer-based variational autoencoders. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 474–489.

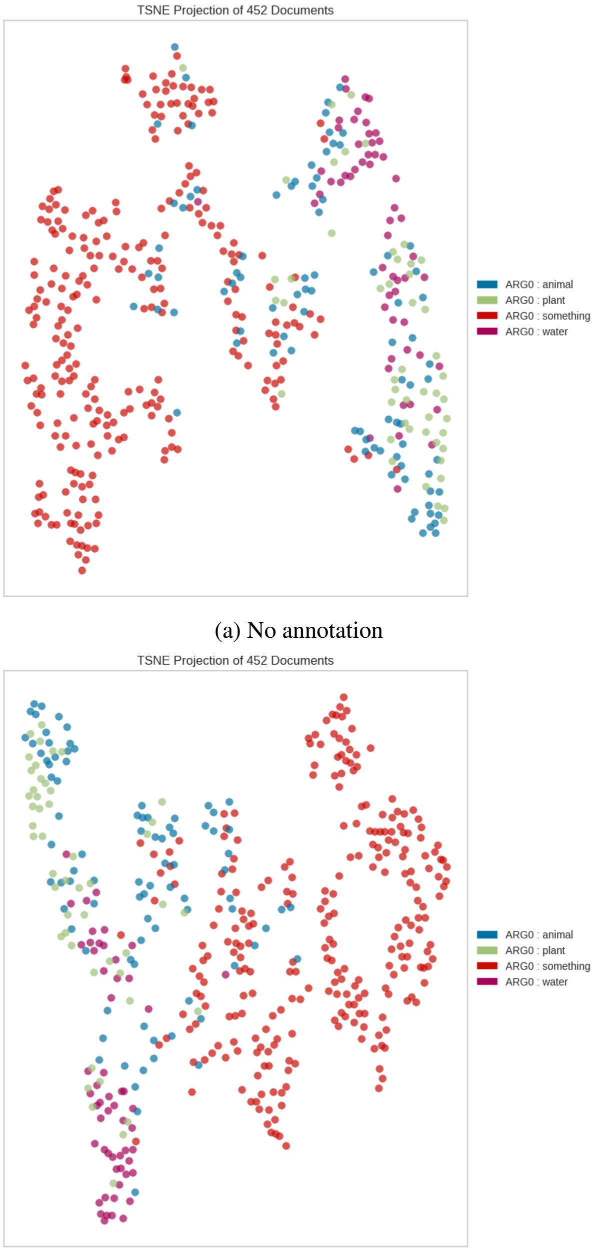
A API examples

A.1 LangVAE

```
# Creates a GPT-2 based decoder expecting
# a latent vector of size 128, that
# generates a maximum of 32 tokens,
# distributed on any number of CUDA GPUs.
decoder = SentenceDecoder(
    "gpt2", latent_size=128,
    max_len=32, device="cuda",
    device_map="auto"
)

# Creates a BERT based encoder producing
# a latent vector of size 128, expecting
# GPT-2 tokenised inputs.
encoder = SentenceEncoder(
    "bert-base-cased", latent_size=128,
    decoder.tokenizer, device="cuda"
)

# Defines a basic VAE model configuration
model_config = VAEConfig(latent_dim=128)

# Initialise LangVAE model
model = LangVAE(
    model_config, encoder, decoder
)

# Alternatively, loads a pretrained
# checkpoint from the HF Hub.
org = "neuro-symbolic-ai"
name="eb-langvae-flan-t5-base-gpt2-l128"
model = LangVAE.load_from_hf_hub(
    f"{org}/{name}"
)
```

B Qualitative results

We present here qualitative results that did not fit in the main text.



(a) No annotation

(b) SRL annotation

Figure 5: TSNE plots for the [bert-base-cased, Llama-3.2-3B] combination, without (a) and with (b) SRL annotated inputs. We can observe a better separation of the *water* and *animal* subjects on the annotated model.

Source	**Target**	**Distance**	**Generate**
the high seas	the continent	0.361	1. the high 2. the high 3. the sea 4. the sea 5. the sea 6. the sea 7. the sea 8. the land 9. the world 10. the world
a primary schooler	a college student	0.299	1. a primary school 2. a primary school 3. a junior school 4. a junior school 5. a high school 6. a high school 7. a high student 8. a college 9. a college 10. a student

Table 2: Interpolation example using a LangVAE model trained on the Wiktionary dataset. Ten points connecting from the source to the target latent vectors are decoded to generate a list of interpolated sentences. We can observe a semantic progression when connecting terms for which there are intermediate senses.