Graph-Induced Syntactic-Semantic Spaces in Transformer-Based Variational AutoEncoders

Yingji Zhang[1†], **Marco Valentino**[2], **Danilo S. Carvalho**[1,3],
Ian Pratt-Hartmann[1], **André Freitas**[1,2,3]
[1] Department of Computer Science, University of Manchester, United Kingdom
[2] Idiap Research Institute, Switzerland
[3] Cancer Biomarker Centre, CRUK Manchester Institute, United Kingdom
[1]{firstname.lastname}@[postgrad.][†]manchester.ac.uk
[2]{firstname.lastname}@idiap.ch

Abstract

The injection of syntactic information in Variational AutoEncoders (VAEs) can result in an overall improvement of performances and generalisation. An effective strategy to achieve such a goal is to separate the encoding of distributional semantic features and syntactic structures into heterogeneous latent spaces via multi-task learning or dual encoder architectures. However, existing works employing such techniques are limited to LSTM-based VAEs. This work investigates latent space separation methods for structural syntactic injection in Transformer-based VAE architectures (i.e., Optimus) through the integration of graph-based models. Our empirical evaluation reveals that the proposed end-to-end VAE architecture can improve theoverall organisation of the latent space, alleviating the information loss occurring in standard VAE setups, and resulting in enhanced performances on language modelling and downstream generation tasks.

1 Introduction

Injecting explicit syntactic information in Variational AutoEncoders (VAEs) (Kingma and Welling, 2013) has led to improved performance on several language generation tasks, such as paraphrasing and translation (Dai et al., 2018; Chen et al., 2017; Felhi et al., 2022; Yang et al., 2021). Among existing techniques, a line of research explores syntactic injection via sentence-level semantics-syntax disentanglement, which consists in the explicit separation of distributional semantic and structural syntactic features through the optimisation of heterogeneous latent spaces (Bao et al., 2019a; Chen et al., 2019; Zhang et al., 2019). Such methods, implemented under multi-task learning or dual encoder architectures, have been demonstrated to improve: (i) generation controllability and interpretability (Bao et al., 2019a; Zhang et al., 2022), (ii) robustness and generalisation, (iii) fine-grained representation and latent space organisation (Chen et al., 2019), and more importantly (iv) injecting syntactic features into VAEs can allow for optimization in low-dimensional and regularized latent Gaussian space, rather than *complex discrete sequence spaces* as investigated in previous work (Pouran Ben Veyseh et al., 2020; Zanzotto et al., 2020; Li et al., 2023; Mohammadshahi and Henderson, 2023), which represents an efficient to improve text generation (Qin et al., 2020; Kumar et al., 2021). However, most of these methods focus on LSTM-based VAEs, and their effectiveness for larger architectures based on Transformers, such as Optimus (Li et al., 2020), is still under-explored.

To combine the benefits of larger pre-trained VAEs and latent separation methods, this paper focuses on the injection of structural syntactic information in Transformer-based VAEs (i.e., Optimus (Li et al., 2020)). Specifically, we investigate a first overarching research question: *"RQ1: How can we best capture explicit syntactic information in the latent space of Transformer-based VAEs?"* we address this question by directly intervening on the Optimus architecture to induce a latent space separation via graph-based (Kipf and Welling, 2016a) and sequential neural encoders (Devlin et al., 2018). Specifically, our hypothesis is that Graph Neural Networks (GNNs) (Kipf and Welling, 2016a; Hamilton et al., 2017; Yun et al., 2020) can induce specialised and complementary latent representations that can better capture structural syntactic relations and alleviate the information bottleneck in VAEs' semantic encoder (Alemi et al., 2016; Tenney et al., 2019) (i.e. trade-off between semantics and syntax).

Subsequently, we focus on the problem of leveraging multiple, specialised latent spaces derived from the dual encoder architecture for decoding. This leads to several challenges (Figure 1) since (i) the syntactic representations may not possess a one-to-one mapping with the semantic representations (i.e., one syntactic structure can correspond

Findings of the Association for Computational Linguistics: NAACL 2024, pages 474–489
June 16-21, 2024 ©2024 Association for Computational Linguistics

to multiple sentence representations), (ii) the optimisation of heterogeneous latent spaces can result in different latent distributions, a feature that can affect decoding and language generation performance, and (iii) compared with an LSTM decoder, Transformer-based decoders (e.g., GPT2) are typically larger and contain information acquired during pre-training, being more difficult to control.

Those challenges lead to our second research question: *"RQ2. How can multiple, specialised latent spaces be effectively injected into the VAE decoder?"* To answer it, we investigate injection mechanisms for Transformer-based VAEs via the following methods: (i) we separately inject syntax and semantic representations into the attention weights of the decoder (i.e., Query and Key-Value), and (ii) consider low-rank injections, including *addition*, *memory* (Li et al., 2020), and *tensor fusion* (Liu et al., 2018), which directly operate over the attention matrices and potentially reduce information redundancy (Hu et al., 2022).

We perform extensive experiments to evaluate the resulting VAE architectures on both mathematical expressions (Valentino et al., 2023; Meadows et al., 2023) and natural language explanatory sentences (Jansen et al., 2018). Overall, our contributions can be summarised as follows: **1.** We propose a dual encoder architecture for Transformer-based VAEs integrating graph-based and sequential models to better capture and disentangle semantic and structural syntactic features in multiple, specialised latent spaces. **2.**We explore the injection of such representations into the decoder of Transformer-based VAEs via low-rank vector operations to better guide the generation process. **3.** We perform extensive experiments showing that the adoption of a graph-based encoder coupled with a transformer encoder can reduce the loss of information in the sentence bottleneck, resulting in improved reconstruction and language modelling. Overall, we found that the proposed VAE architecture can significantly improve performance and generalisation when compared to sentence-level VAE baselines. Our complete experimental code is available online to encourage future work in the field[1].

2 Preliminaries

Latent Space Injection. In Optimus, the transformation between latent (i.e., Gaussian) and observed (i.e., generated sentences) spaces can be done by intervening on the Key-Value attention weights of the decoder (i.e., GPT2) via *memory* injection (Li et al., 2020). Specifically, the latent representation z produced by the encoder (i.e., BERT) is concatenated into the original Key-Value weights of GPT2 as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{Q[z; K]^T}{\sqrt{d}})[z; V]$$

Where Q has dimension $\mathbb{R}^{64\times \text{seq}}$, and $[z; K], [z; V]$ have dimension $\mathbb{R}^{64\times(\text{seq}+1)}$ (where 64 is the dimension of GPT2 attention, seq is sequence length). In order words, the decoder model is explicitly guided in the generation process by conditioning KV on z. In this work, however, we focus on heterogeneous representations encoding distributional semantic and structural syntactic features in separate latent spaces (see Figure 1). Such a separation requires going beyond the *memory* injection setup and developing different methods to effectively condition the decoding process in GPT2.

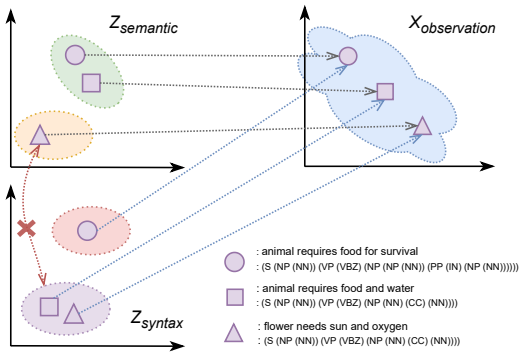

Figure 1: Decoding under heterogeneous syntactic-semantic spaces can result in two main challenges: (i) The syntactic representations may not possess a one-to-one mapping with the semantic representations (i.e., one syntactic structure can correspond to multiple sentence representations), (ii) the optimisation of heterogeneous latent spaces can result in different latent distributions, making generation hard to control.

Semantic-Syntax Relation. Following the *principle of compositionality*, the semantics of a sentence can be seen as a composition of word-level semantics, induced by the meaning of individual words and their relations (Dowty et al., 2012; Yao et al., 2023). Instead of considering sentence-level semantics only as a composition of word content as done in previous work (Bao et al., 2019a), this work uses the notion of sentence semantics as word content plus positional elements (i.e. *word order*

[1] https://github.com/SnowYJ/sem_syn_separation

typology (Sankaravelayuthan, 2020)), which has been well captured by Transformer-based encoders. Under this constraint, mutual information naturally exists between semantics and syntax. Therefore, although separating semantic and syntactic features in heterogeneous latent spaces can lead to representations that are not geometrically aligned in the Gaussian space (Figure 1), such mutual information can be captured through low-rank injection (Zhang et al., 2019), which directly work on QKV instead of token embeddings or the last hidden representation (Hu et al., 2022).

3 Methodology

Our methodology consists of two main phases. First, we investigate different encoding strategies to explicitly capture syntactic and structural information in a separate latent space. Subsequently, we explore techniques to fuse syntactic and semantic features and inject them into the decoder model. Regarding the encoding phase, we explore four architectures based on two different configurations (i.e., *multi-task learning* and *dual encoder*) integrating both *sequential* and *graph-based* models under Optimus (BERT-GPT2) *memory* setup (see Figure 6). Regarding the decoding phase, we consider the best encoding configuration in terms of syntactic representation and propose different injection mechanisms via low-rank operations over attention-weight matrices of GPT2.

The following sections describe each phase in detail (Sections 3.1 and 3.2), including how the encoding and decoding stages are integrated into an end-to-end VAE architecture (Section 3.3).

3.1 Encoding Syntactic-Semantic Spaces

Multi-Task Learning. Bao et al. (2019a) proposed a multi-task learning strategy to achieve such a goal in LSTM-based VAEs via learning and fusing two distinct latent representations. They adopt a separate space for encoding explicit syntactic dependencies through the adoption of an LSTM decoder used to reconstruct flattened constituency parse trees. Here, we build upon this setup to enrich the latent representation in Optimus (Li et al., 2020). Specifically, given a separate latent syntax representation, z_{syn}, encoded via BERT (Devlin et al., 2018), we explore the following mechanisms (see Figure 6): **1.** Similarly to (Bao et al., 2019b), we adopt an LSTM (Hochreiter and Schmidhuber, 1997) decoder to generate linearised syntactic trees, where z_{syn} is fed into the first hidden state of the LSTM. We refer to this configuration as *Optimus (LSTM)*. **2.** We jointly train a Variational Graph AutoEncoder (VGAE, Kipf and Welling (2016b)) on syntactic trees, where the latent node embeddings are mean-pooled into a sentence-level syntax representation z^{gcn}_{syn}. We refer to this configuration as *Optimus (VGAE)*. Here, the syntactic representations z^{gcn}_{syn} and z_{syn} can be optimized via MSE in a multi-task setting. Specifically, the general objective function can be formalised as:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi(z_{sem}, z_{syn}|x)}\Big[\log p_\theta(x|z_{sem}, z_{syn})\Big] - \text{KL}(\phi(z_{sem}|x)||p(z)) - \text{KL}(\phi(z_{syn}|x)||p(z)) + \mathcal{L}_{\text{syn}}(z_{syn})$$

Where q_ϕ, p_θ represent the encoder and decoder. The objective functions for optimising the syntactic spaces $\mathcal{L}_{\text{syn}}(z_{syn})$ can be specialised according to the model: LSTM: $\mathcal{L}^{lstm}_{\text{syn}}(z_{syn}) = \sum_{i=1}^{n} \log p(s_i|s_1, \ldots, s_{i-1}, z_{syn})$ and VGAE: $\mathcal{L}^{vgae}_{\text{syn}}(z_{syn}) = \sum_{j=1}^{dim}(z^j_{gcn} - z^j_{syn})^2 + \mathcal{L}^{vgae}(A, N)$ Where s_i represents the token of a flattened syntax tree, while A and N are the Adjacent matrix and Node embeddings of the syntax tree. Additional details for the VGAE model and the optimisation of $\mathcal{L}^{vgae}$ can be found in the original paper (Kipf and Welling, 2016b).

Dual Encoder. In addition to the multi-task learning setup, we build upon Zhang et al. (2019); Huang and Chang (2021) which propose two distinct language encoders to induce syntactic disentanglement. Specifically, we experiment with:

1. Two distinct BERT encoders via a Siamese neural network. We refer to this configuration as *Optimus (Siam)*. **2.** A Graph encoder, such as GCN (Kipf and Welling, 2016a), GraphSAGE (Hamilton et al., 2017), and Graph Transformer (TransCONV, Yun et al. (2020)), coupled with a BERT encoder. We refer to this configuration as *Optimus (GraphEncoder)*. Here, the general objective function can be formalised as:

$$\mathbb{E}_{q^{sem}_\phi(z_{sem}|x), q^{syn}_\phi(z_{syn}|x_{syn})}\Big[\log p_\theta(x|z_{sem}, z_{syn})\Big] - \text{KL}(\phi(z_{sem}|x)||p(z)) - \text{KL}(\phi(z_{syn}|x)||p(z))$$

Where $q^{sem}_\phi, q^{syn}_\phi$ represent semantic and syntax encoders respectively, while x_{syn} represents the input for the syntax encoder. For graph encoders, we represent x_{syn} using an adjacency matrix and node embedding pairs. For the language syntax

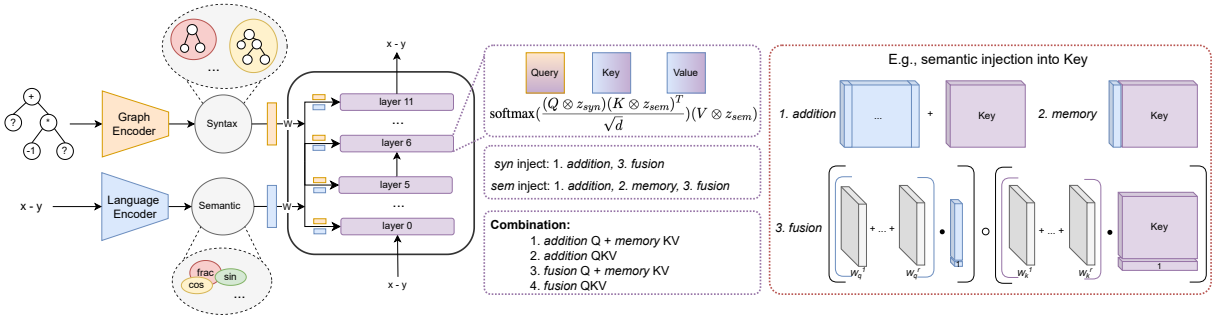

Figure 2: Architectural overview. Semantic and syntactic features are encoded into heterogeneous latent spaces via graph-based and sequential encoders. The resulting latent spaces are injected into the GPT2 decoder via low-rank operations.

encoder, on the other side, we represent x_{syn} as a flattened syntactic tree without word content.

As our experiments revealed that the *dual graph-sequential encoder* configuration (i.e., *Optimus (GraphEncoder)*) can achieve the best results in terms of syntactic representation (see Table 1), we consider this setup for integration into an end-to-end VAE architecture (see Section 3.3).

3.2 Decoding Heterogeneous Representations

To preserve the separation of the latent spaces and, at the same time, leverage heterogeneous representations during decoding, we explore methods to inject semantic (i.e., z_{sem}) and syntactic space (i.e., z_{syn}) directly into the attention mechanism of GPT2 (via QKV). Specifically, we inject different latent representations to different attention weights:

$$\text{softmax}(\frac{(Q \otimes z_{syn})(K \otimes z_{sem})^T}{\sqrt{d}})(V \otimes z_{sem})$$

Where $\otimes$ represents the latent injection operation. As for syntactic injection (z_{syn}), we consider two kinds of low-rank operations $\otimes$, *addition*, and *fusion* (Liu et al., 2018), which directly work on attention weights. As for *addition*, we inject z_{syn} into each low-rank token representation in Q, which can be formalised as follows: $\tilde{Q} = \sum_{i=1}^{seq} Q[i,:] + z_{syn}$ Where $\tilde{Q}$ represents the new Q values obtained after syntax injection. As for *fusion*, we adapt the tensor fuse operation (Liu et al., 2018; Hu et al., 2022). In more detail, given a hyper-parameter, rank $r = 4$, the $\tilde{Q}$ can be described as: $\tilde{Q} = (\sum_{i=1}^{r} W_q^i[Q; \mathbb{1}]) \circ (\sum_{i=1}^{r} W_z^{i,syn}[z_{syn}; \mathbb{1}])$ Where $\mathbb{1}$ is the matrix of ones, $W_z^{i,syn}$ and W_q are the trainable linear transformations.

As for semantic injection (z_{sem}), we consider three operations: *addition*, *memory*, and *fusion*, where *addition* and *fusion* operations are the same as before but works on KV. *Memory* is the same as Optimus *memory* injection (Li et al., 2020) as we described in section 2. We refer (Liu et al., 2018) for an in-depth description of tensor fusion.

3.3 VAE Architecture

Encoder. At the encoding stage, we consider the dual graph-sequential encoding mechanism adopting BERT as a sequential encoder and experimenting with two different graph-based encoders, including GraphSAGE (Hamilton et al., 2017), and Graph Transformer (TransCONV, Yun et al. (2020)). The dual graph-sequential encoding can alleviate the information bottleneck derived from the encoding stage (as illustrated in 3.3). To derive the syntactic space, z_{syn}, we use a mean pooling operation to obtain a sentence-level representation from the node embeddings N and the adjacency matrix A: $\text{Embed}_{syn} = \text{MeanPool}(\text{GraphEnc}(A, N))$

For the semantic space, z_{sem}, we consider the special token [CLS] in BERT as the input of a linear transformation (W) to obtain a sentence-level representation: $\text{Embed}_{sem} = W(\text{LanguageEnc}(x)_{[\text{CLS}]})$ Where x is the input sentence. Both spaces are constrained to follow a Gaussian distribution by learning the parameters μ and σ through multilayer perceptrons W_μ^{sem}, W_σ^{sem}, W_μ^{syn}, and W_σ^{syn}. The final latent representations can be obtained via: $z_{sem(syn)} = W_\mu^{sem(syn)} \times \text{Embed}_{sem(syn)} + W_\sigma^{sem(syn)}$

Decoder. Since the architecture constraint, z_{sem} and z_{syn} have the potential to capture diverse features with a high level of disentanglement. To this end, we experiment with different decoding injection setups and low-rank operations (Section 3.2) : (1) *addition* for QKV (i.e., addition QKV), (2) *fusion* for QKV (fusion QKV), (3) *addition* for Q

and *memory* for KV (addition Q), and (4) *fusion* for Q and *memory* for KV (fusion Q).

Optimisation. Our model can be trained via Evidence Lower Bound (ELBO) x (Kingma and Welling, 2013). To avoid the KL vanishing issue, which refers to the Kullback-Leibler (KL) divergence term in the ELBO becomes very small or approaches zero, we select the cyclical schedule to increase weights of KL β from 0 to 1 (Fu et al., 2019) and a KL thresholding scheme (Li et al., 2019) that chooses the maximum between KL and threshold λ. The final objective function can be described as follows:

$$\mathcal{L}_{\text{VAE}} = \mathbb{E}_{q_\phi^{sem}(z_{sem}|x), q_\phi^{syn}(z_{syn}|A,N)} \Big[\log p_\theta(x|z_{sem}, z_{syn}) \Big] - \beta \max \left[\lambda, \text{KL} q_\phi^{sem}(z_{sem}|x) || p(z) \right] - \beta \max \left[\lambda, \text{KL} q_\phi^{syn}(z_{syn}|x) || p(z) \right]$$

Information Bottleneck The dual graph-sequential encoding setup has the potential to alleviate information bottlenecks for sentence representations. In detail, Li et al. (2020) revealed that $\mathcal{L}_{\text{VAE}}$ is the upper bound of the information bottleneck (IB) (*information bottleneck principle*, Tishby et al. (2000)).

$$\mathcal{L}_{\text{VAE}} \geq (1-\beta) I_q(s,z) = \mathcal{L}_{\text{IB}}^{BERT}$$

where s and z represent sentence and its corresponding latent representation z, I_q is the mutual information, q is encoder, $\mathcal{L}_{\text{IB}}$ is the Lagrange relaxation form (Tishby et al., 2000). As we mentioned in section 2, s is composed of two kinds of information $\{x_{sem}\}$ and $\{x_{syn}\}$. In vanilla Optimus, $I(s,z)$ can be expanded into:

$$I_q(s,z) = I_q(x_{sem} + x_{syn}; z) = I_q(x_{sem}, z) + I_q(x_{syn}, z) - I_q(x_{sem}, x_{syn}|z)$$

Similarly, under the dual graph-sequential encoder setup, the mutual information can be described as:

$$\mathcal{L}_{\text{IB}}^{BERT-graph} = I'_q(s,z) = I_q(x_{sem}, z) + I_q(x_{syn}, z)$$

As we claimed before, $\{x_{sem}\} \cap \{x_{syn}\} \neq \emptyset$. Therefore, $\mathcal{L}_{\text{IB}}^{BERT} - \mathcal{L}_{\text{IB}}^{BERT-graph} = I_q(s,z) - I'_q(s,z) = -I_q(x_{sem}, x_{syn}|z) < 0$, indicating that the separated encoders can alleviate the information bottleneck.

4 Empirical Evaluation

Following the stages in our methodology, we first evaluate different encoding setups for injecting syntactic information into VAEs (Section 3.1). Subsequently, we consider the best encoding configuration to examine which decoding strategy (Section 3.3) can lead to better language modelling performances. Finally, we evaluate the best architectural setup for downstream tasks. To experiment, we focus on both *explanatory sentences* and *mathematical expressions*. The rationale behind this choice is that (1) explanatory sentences (Jansen et al., 2018; Valentino et al., 2022; Thayaparan et al., 2021; Zhang et al., 2023b) provide a semantically challenging yet sufficiently well-scoped scenario to evaluate the syntactic and semantic organisation of the space; (2) mathematical expressions (Valentino et al., 2023; Meadows et al., 2023) follow a well-defined syntactic structure and set of symbolic rules that are notoriously difficult for neural models. All experimental details are provided in Appendix A.

4.1 Encoding: Latent Representations

Evaluation. Firstly, we evaluate different encoding setups to the effect of semantic-syntax distribution in latent space from three perspectives: (i) latent space geometry: whether the latent space can capture the corresponding features – i.e., sentences with the same/different features are clustered/separated accordingly in the latent space. In this case, we can evaluate the organisation of the latent space via MSE of k-mean (Zhang et al., 2022, 2023a; Michlo et al., 2023); (ii) syntactic features: following the probing method (Conneau et al., 2018), we train a linear classifier to predict tree depth. Here, better classification performances indicate a higher separability of syntactic features in the latent space; and (iii) semantic and syntax space alignment: we adopt statistical metrics to compare latent distributions such as Mutual Information (MI), Kullback–Leibler divergence (KL), and Wasserstein distance (Wass). As illustrated in Table 1, we can observe that (1) the Optimus(GraphEncoder) can better capture the syntactic structures and induce a better latent space separation, (2) It can lead to a better organisation of the semantic space MSE(sem). We will further explore this phenomenon in subsequent sections.

Visualisation. Next, we visualize the cluster separation of latent space via t-SNE (van der Maaten

Corpus	*Mathematical expression*				*Explanatory sentences*				
Proxy metrics	MSE(sem)↓	MSE(syn)↓	Acc_{dep}(syn)↑	Acc_{dep}(sem)↓	MSE(sem)↓	MSE(syn)↓	Acc_{dep}(syn)↑	Acc_{dep}(sem)↓	$F1_{dep}$(sem)↓
LSTM	079.02	070.48	000.74	000.74	176.39	158.03	000.40	000.40	000.41
VGAE	125.68	434.52	000.81	000.82	169.42	110.30	000.40	000.38	000.45
Siam	191.97	053.90	000.85	000.52	074.86	031.95	000.43	000.35	000.42
GraphEncoder	–	–	–	–	–	–	–	–	–
+ GCN	**004.31**	065.79	000.72	**000.27**	069.77	091.94	000.49	**000.12**	**000.30**
+ GraphSAGE	208.21	053.20	**000.98**	000.52	058.12	004.10	000.50	000.39	000.46
+ TransConv	249.00	**038.30**	**000.98**	000.57	**058.10**	**003.35**	**000.51**	000.38	000.47

$F1^{*}_{dep}$(sem)↓	$F1_{dep}$(syn)↑	MI(sem,syn)↓	KL(sem‖syn)↑	Wass(sem,syn)↑	$F1_{dep}$(syn)↑	MI(sem,syn)↓	KL(sem‖syn)↑	Wass(sem,syn)↑
000.71	000.70	004.88	005.74	000.53	000.43	004.87	001.01	000.78
000.84	000.84	004.85	026.12	000.32	000.44	004.66	007.04	000.90
000.41	000.87	004.85	011.95	000.69	000.44	004.96	008.72	000.80
–	–	–	–	–	–	–	–	–
000.24	000.79	004.82	024.05	000.72	**000.54**	004.78	011.77	000.30
000.42	**000.98**	005.04	005.12	000.69	000.44	004.45	**043.45**	**001.92**
000.52	**000.98**	**004.80**	**031.63**	**001.19**	000.48	**003.54**	012.78	000.75

Table 1: Proxy metrics for evaluating the organisation of the latent syntactic and semantic space for different encoding configurations of Optimus. The **best** results indicate that the graph-language encoding setup can effectively capture syntactic information and maintain separation.

Corpus	*Mathematical expression*										*Explanatory sentences*				
Metrics	EVAL		VAR-SWAP		EASY		EQ-CONV		LEN		BLEU	BLEURT	Cosine	Loss↓	PPL↓
sentence VAE baselines															
01. AAE(768)	0.10	0.75	0.00	0.25	0.02	0.53	0.00	0.54	0.00	0.51	0.35	-0.95	0.80	3.35	28.50
02. LAAE(768)	0.00	0.43	0.00	0.25	0.00	0.27	0.00	0.29	0.00	0.44	0.26	-1.07	0.78	3.71	40.85
03. DAAE(768)	0.00	0.24	0.00	0.21	0.00	0.21	0.00	0.22	0.00	0.42	0.22	-1.26	0.76	4.00	54.59
04. β-VAE(768)	0.00	0.14	0.00	0.15	0.00	0.13	0.00	0.14	0.00	0.35	0.06	-1.14	0.77	3.69	40.04
05. Optimus(768)	0.99	0.99	0.00	**0.38**	0.81	0.93	0.00	0.81	**0.14**	0.76	0.35	-0.59	0.83	0.98	2.66
different encoding setups with memory injection															
06. LSTM	**1.00**	**1.00**	0.00	0.35	0.73	0.94	0.00	0.77	0.06	0.74	0.41	-0.41	0.85	1.04	2.82
07. VGAE	0.98	0.99	0.00	0.34	0.72	0.93	0.00	0.74	0.04	0.71	0.26	-0.91	0.78	1.14	2.55
08. Siam	**1.00**	**1.00**	0.00	0.30	0.22	0.80	0.00	0.78	0.03	0.75	0.49	-0.15	0.88	0.94	2.55
GraphEncoder															
09. + GCN	0.00	0.40	0.00	0.22	0.00	0.27	0.00	0.37	0.00	0.43	0.15	-1.19	0.75	1.24	3.45
10. + GraphSAGE	0.88	0.96	0.00	0.28	0.06	0.46	0.00	0.69	0.00	0.60	0.45	-0.28	0.87	1.00	2.71
11. + TransCONV	0.89	0.95	0.00	0.28	0.14	0.53	0.00	0.67	0.00	0.61	0.17	-1.16	0.75	1.21	3.35
Graph-language encoders: injecting syntax into Q, semantic into KV															
BERT-GraphSAGE															
12. + addition Q	0.99	0.99	0.00	0.27	0.23	0.63	0.00	0.71	0.02	0.66	0.60	0.22	0.92	0.74	2.09
13. + addition QKV	**1.00**	**1.00**	0.00	0.35	0.65	0.90	0.00	0.80	0.06	0.75	0.63	0.31	0.93	0.65	1.91
14. + fusion Q	0.94	0.97	0.00	0.29	0.08	0.63	0.00	0.71	0.00	0.62	0.55	0.03	0.91	0.90	2.45
15. + fusion QKV	**1.00**	**1.00**	0.00	**0.38**	0.37	0.84	0.00	0.80	0.02	0.73	0.46	-0.23	0.88	1.10	3.00
BERT-TransCONV															
16. + addition Q	0.98	0.99	0.00	0.26	0.31	0.69	0.00	0.67	0.01	0.63	0.59	0.18	0.92	0.76	2.13
17. + addition QKV	**1.00**	**1.00**	0.00	**0.38**	**0.90**	**0.98**	0.00	**0.82**	0.10	**0.78**	**0.65**	**0.35**	**0.94**	**0.62**	**1.85**
18. + fusion Q	0.96	0.98	0.00	0.29	0.18	0.60	0.00	0.74	0.00	0.64	0.53	-0.02	0.90	0.98	2.66
19. + fusion QKV	0.99	0.99	0.00	0.35	0.45	0.82	0.00	0.80	0.01	0.74	0.46	-0.16	0.88	1.13	3.09

Table 2: Results on language modelling. Regarding mathematical expressions, we adopt exact match (left) and bleu (right) as evaluation metrics for each test set. The best results are highlighted in **blue**.

and Hinton, 2008) (see Figure 3). From the visualisation, we can observe that the Optimus injection with a separated GraphEncoder can induce a better separation between different syntactic clusters. We also provide a qualitative evaluation by decoding the latent representation of each cluster (Table 5, 6, and 7) and visualisation for explanatory sentences (Figure 7, 8, and 9) in Appendix B. These results reveal that the integration of graph-based and sequential models in a dual-encoder setup can better capture structural syntactic information while maintaining a separation between latent spaces.

4.2 Decoding: Language Modelling

Baselines. We assess performances on language modelling using a different set of baselines[2]. Specifically, we evaluate the performance of vanilla Optimus (Li et al., 2020) and four LSTM-based autoencoders (AEs), including β-VAE (Higgins et al., 2016), adversarial AE (Makhzani et al.

[2]We choose the standard transformer-based VAE (Optimus) with single latent space (i.e., with the prior being a standard Gaussian distribution) for a fair comparison. Some variants, such as Della (Hu et al., 2022), DPrior (Fang et al., 2022), (Li et al., 2022), etc., were not selected.

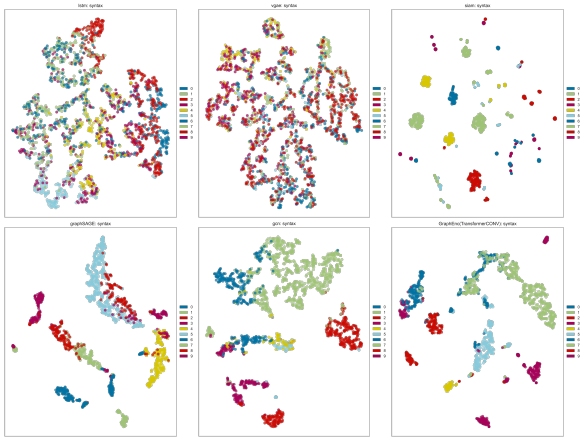

Figure 3: Visualizing the syntactic clusters for mathematical expressions reveals that graph encoders can better represent syntactic information in latent space (top: LSTM, VGAE, Siam, bottom: graph encoders with GraphSAGE, GCN, TransformerCONV).

(2016), AAE), label adversarial AE (Rubenstein et al. (2018), LAAE), and denoising adversarial autoencoder (Shen et al. (2020), DAAE). All baselines have a latent size of 768. For semantic-syntax separated VAE setups, we evenly split the latent space for both. Moreover, we compare the proposed injection mechanism via low-rank operations with a standard memory injection (Li et al., 2020).

Metrics. As for mathematical latex expressions, we use Exact Match and Bleu to evaluate the robustness of models on five different test sets, where four of them include out-of-distribution examples, (1) EVAL: mathematical expressions following the training set's distribution (like $U + cos(n)$), (2) VAR: mathematical expressions composed of a different set of variables (like $U + cos(beta)$), (3) EASY: mathematical expressions with a lower number of variables (like $cos(n)$), (4) EQ: mathematical derivations with equality insertions (like $E = U + cos(n)$), (5) LEN: mathematical derivations with a higher number of variables (like $U + cos(n)) + A + B$). For explanatory sentences, we use five metrics, including BLEU (Papineni et al., 2002), BLEURT (Sellam et al., 2020), cosine similarity from pre-trained sentence T5 (Ni et al., 2021), cross-entropy (Loss), and perplexity (PPL).

Results. Firstly, we evaluate the performance of baselines with different syntactic injection setups. In the middle of Table 2, most configurations lead to lower performance, especially when using graph encoders, compared to vanilla Optimus, indicating that a standard memory injection mechanism for leveraging heterogeneous latent space is not effective. Conversely, by comparing line 05 to lines 12,14,16,18, injecting only syntactic information in Q can improve reconstruction performances on explanatory sentences. Moreover, we evaluate whether injecting heterogeneous latent representations into different attention components (Q,K,V) can further improve the results. In the bottom of Table 2, injecting semantic and syntax spaces into different attention components can additionally improve model performance (lines 9-11 vs 12,14,16,18), demonstrating that semantic and syntax space possess complementary features. Finally, we evaluate which injection strategies can achieve the best results. We found that *addition* injection with BERT-TransCONV (line 17) can achieve the best overall results on both corpora. Next, we further analyse why syntax injection can improve model performance in natural language sentences.

Analysis. Under the VAE setup, we conjecture that the syntax and semantics separation allows the BERT encoder to capture and represent more fine-grained lexical information, alleviating the information loss in the sentence bottleneck. We provide a set of qualitative examples in Table 8. Given an input: *a bee is a kind of living thing*, we found the reconstruction of vanilla Optimus to be *a frog is a kind of amphibian*. This shows that Optimus is distracted by syntactic features, (*x is a kind of y*) that are highly frequent in the training set and struggles in the reconstruction of specific lexical content (i.e., *frog* and *amphibian*). In contrast, the proposed architecture allows the semantic space to specialise in lexical content since the graph encoder already captures the syntax. To additionally support our claim, we visualize the attention weights of GPT2. In figure 4, the first column of each heatmap represents the lexical information carried by the latent representation. We can observe that the proposed architecture with BERT-TransCONV + *addition Q* setup (right) pays more attention to specific lexical elements (i.e., *bee*) compared to vanilla Optimus (left). This also explains how the integration of a graph-based encoder can indirectly improve organisation for the semantic space (MSE in Table 1). We provide additional heatmaps in Appendix B.

4.3 Downstream Evaluation

Guided Generation. One advantage of the VAE architecture is that it allows controlling sentence

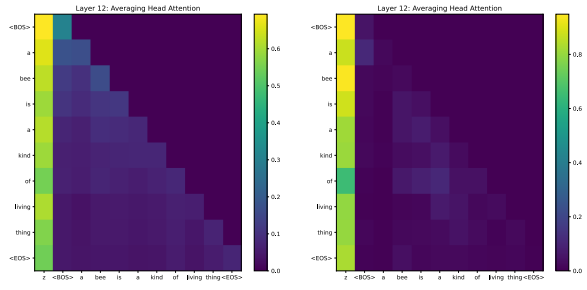

Figure 4: Visualizing attention weighs (left: vanilla Optimus, right: BERT-TransCONV with *addition Q* setup) where *bee*: 0.58 < 0.94, *living thing*: (0.27, 0.15) < (0.80, 0.80).

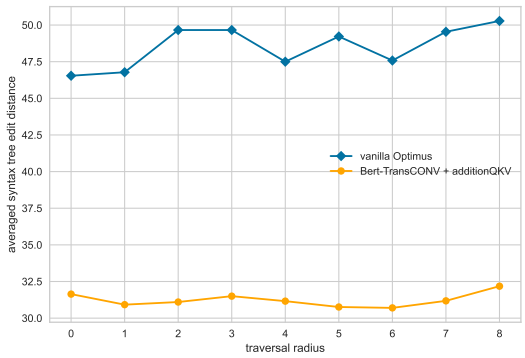

Figure 5: Traversing semantic space with increasing traversal radius while keeping syntax space fixed. We can observe improved syntax control in decoding by separating syntax and semantics.

generation by manipulating latent representations via traversal and interpolation. In this experiment, we quantitatively assess the controllability of the decoding via latent traversal. Specifically, given an initial point in Gaussian space, we perform an *Ornstein-Uhlenbeck* random walk (Pinsky and Karlin, 2010) [3] for semantic space and fix syntax space. In detail:

1. We set the traversal radius (r) - a predefined hyper-parameter, and sample an initial point/vector (sampled from Gaussian space).

2. We traverse the semantic latent space using *Ornstein-Uhlenbeck* random walk and calculating the Euclidean distance between the traversed vectors and the initial point.

3. We keep only the samples whose distance is > r_{t-1} and < r_t when $t = 1, r_{t-1} = 0$.

4. We generate the sentences from the latent spaces using the model and then compute the syntax tree edit distance (i.e., the distance between the syntactic trees) of the samples retrieved in step 3 and calculate the average distance.

5. Repeat 2 - 5.

If the model can learn semantic-syntactic separation, the generated sentence can be syntactically controlled. To experiment, we quantitatively evaluate the similarity of syntactic structures between initial and traversed sentences via syntax tree edit distance. We gradually increase the radius of the random walk to perform a comparison between vanilla Optimus and BERT-TransCONV(addition QKV). In Figure 5, we can conclude that the proposed architecture can better hold the syntax structure, indicating better separation. We provide qualitative examples of such behaviour in Appendix B.

[3] $\tilde{z}_{t+1} = -\gamma\tilde{z}_t + \sigma W_t$ where t is the index, $W_t \in N(0, 1)$, γ and σ are scalar hyper-parameters.

Mathematical Derivations. Finally, we explore the representation quality for mathematical expressions on downstream equational inference tasks (Meadows et al., 2023). Specifically, we focus on expression derivation, where, given a premise x and a mathematical operation t (i.e., differentiation, integration) the goal is to predict whether a target mathematical expression y can be derived from x via t. Here, we adopt the dataset from (Valentino et al., 2023) and examine whether a linear probing classifier (Ferreira et al., 2021) trained on latent expression representations encoded from frozen pre-trained models can predict the correct operation t in a multi-label classification problem (i.e., given premise x and target result y) and whether the classifier can predict a valid conclusion y (i.e. Conclusion Classification) given a premise x in a binary classification setting (using random negative examples). Experimental results reveal that separately injecting latent semantic and syntactic representations can provide complementary information and improve performance on both tasks.

5 Related work

Language VAE. Most previous language VAE works are based on LSTM instantiated on different generation tasks, including dialogue generation (Zhao et al., 2017), text style transfer (John et al., 2019; Shen et al., 2020), text paraphrasing (Bao et al., 2019a), etc. The development of Optimus (Li et al., 2020) led to more research focusing on how to control the generation of Transformer-based architectures by latent space geometry (Zhang et al., 2022, 2023a) or pre-defined priors (Fang et al., 2022; Li et al., 2022; Hu and Li, 2021). Compara-

Inference Type	Operation Class.		Conclusion Class.	
Metrics	Acc	F1	Acc	F1
Optimus(768)	0.89	0.89	0.68	0.68
LSTM	0.89	0.89	0.59	0.62
VGAE	0.79	0.80	0.56	0.62
Siam	**0.92**	**0.92**	0.59	0.59
GraphEncoder				
+ GCN	0.73	0.74	0.57	0.55
+ GraphSAGE	0.87	0.87	0.64	0.63
+ TransCONV	0.88	0.89	0.63	0.62
Bert-GraphSAGE				
+ addition QKV	0.88	0.88	0.69	0.69
+ fusion QKV	0.90	0.90	**0.71**	**0.71**
Bert-TransCONV				
+ addition QKV	**0.92**	**0.92**	0.68	0.68
+ fusion QKV	0.91	0.91	0.59	0.59

Table 3: Results for the mathematical derivations probing task reveal that separately injecting latent semantic and syntactic representations can provide complementary information, resulting in enhanced performance.

tively, we focused on the semantic-syntax separation with the help of a graph-based encoder. To our knowledge, the combination of graph encoders and VAEs for text generation is underexplored.

Learning Syntactic Representations. From the perspective of model architecture, three kinds of encoders can learn syntactic representations, including graph-based encoders (Wu et al., 2023), sequential encoders (i.e., LSTM (Hochreiter and Schmidhuber, 1997) and Transformers (Vaswani et al., 2017)), and tree-based encoders (Harer et al., 2019) (i.e., using Recursive Neural Networks (Harer et al., 2019; Mrini et al., 2021)), with the latter two commonly used in the natural language generation domain (Raffel et al., 2020). Nevertheless, whether these models truly capture structural information or just the lexical combination of tokens is not fully clarified (Shi et al., 2016). This work uses graph-based encoders (Kipf and Welling, 2016a) to better capture topological relations in syntactic trees. Graph Neural Networks (Zhou et al., 2020) have been effective for encoding syntactic and relational structures in various NLP tasks (Wu et al., 2023; Sachan et al., 2021; Veyseh et al., 2020).

6 Conclusion and Future Work

This work focused on the semantic-syntax separation through language VAEs, especially Optimus (Bert-GPT2), architecture. We first implement several encoding baselines and reveal that language-graph encoding setups can better capture syntax information and maintain semantic-syntax separation. However, the language-graph encoding setup leads to low reconstruction performance. To solve this problem, we explored the integration of heterogeneous latent spaces via injection mechanisms. Experimental results showed that our setup can greatly improve language modelling performance, and revealed that the semantic-syntax separation can assist the language modelling task since the language encoder pays more attention to fine-grained lexical semantics, avoiding the distraction of syntax information captured by the separated syntax encoder, which can alleviate the information bottleneck of the language encoder. In the future, we will investigate graph-to-text generation through VAEs for bridging structural and distributional semantics via latent Gaussian space. By learning the structural semantics distribution as approximated posterior, this type of representation can shorten the gap between deep latent semantics and formal linguistic representations (Banarescu et al., 2013; Mitchell, 2023), integrating the flexibility of distributional-neural models with the properties of linguistically grounded representations, facilitating both interpretability and generative control.

7 Limitations

Although the semantic-syntax separated latent space can provide better latent space geometry, it is still challenging to efficiently control the decoding stage through latent geometry itself, due to the discrete nature of the latent sentence space. Besides, robustness towards out-of-distribution generalization for individual latent spaces has not been investigated and has been left for future work. Finally, while our work revealed that structural syntactic information can be well captured and represented in separated latent spaces, whether such a mechanism can contribute to the representation of explicit structural semantic information as well (i.e., semantic role labels) is not explored in this work.

Acknowledgements

We appreciate the reviewers for their insightful comments and suggestions. This work was partially funded by the EPSRC grant EP/T026995/1 entitled "EnnCore: End-to-End Conceptual Guarding of Neural Architectures" under Security for all in an AI enabled society, by the Swiss National Science Foundation (SNSF) project NeuMath (200021_204617), by the CRUK National Biomarker Centre, and supported by the Manchester Experimental Cancer Medicine Centre and the NIHR Manchester Biomedical Research Centre.

References

Alexander A Alemi, Ian Fischer, Joshua V Dillon, and Kevin Murphy. 2016. Deep variational information bottleneck. *arXiv preprint arXiv:1612.00410*.

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract meaning representation for sembanking. In *Proceedings of the 7th linguistic annotation workshop and interoperability with discourse*, pages 178–186.

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

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

Huadong Chen, Shujian Huang, David Chiang, and Jiajun Chen. 2017. Improved neural machine translation with a syntax-aware encoder and decoder. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1936–1945, Vancouver, Canada. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. A multi-task approach for disentangling syntax and semantics in sentence representations. 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 2453–2464, Minneapolis, Minnesota. Association for Computational Linguistics.

Alexis Conneau, German Kruszewski, Guillaume Lample, Loïc Barrault, and Marco Baroni. 2018. What you can cram into a single $&!#* vector: Probing sentence embeddings for linguistic properties. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2126–2136, Melbourne, Australia. Association for Computational Linguistics.

Hanjun Dai, Yingtao Tian, Bo Dai, Steven Skiena, and Le Song. 2018. Syntax-directed variational autoencoder for structured data. *arXiv preprint arXiv:1802.08786*.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees.

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

David R Dowty, Robert Wall, and Stanley Peters. 2012. *Introduction to Montague semantics*, volume 11. Springer Science & Business Media.

Xianghong Fang, Jian Li, Lifeng Shang, Xin Jiang, Qun Liu, and Dit-Yan Yeung. 2022. Controlled text generation using dictionary prior in variational autoencoders. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 97–111, Dublin, Ireland. Association for Computational Linguistics.

Ghazi Felhi, Joseph Le Roux, and Djamé Seddah. 2022. Exploiting inductive bias in transformers for unsupervised disentanglement of syntax and semantics with VAEs. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5763–5776, Seattle, United States. Association for Computational Linguistics.

Deborah Ferreira, Julia Rozanova, Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2021. Does my representation capture x? probe-ably. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing: System Demonstrations*, pages 194–201.

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

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. 2018. Allennlp: A deep semantic natural language processing platform. *arXiv preprint arXiv:1803.07640*.

Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30.

Jacob Harer, Chris Reale, and Peter Chin. 2019. Tree-transformer: A transformer-based method for correction of tree-structured data. *arXiv preprint arXiv:1908.00449*.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher P. Burgess, Xavier Glorot, Matthew M. Botvinick, Shakir Mohamed, and Alexander Lerchner. 2016. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International Conference on Learning Representations*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Jinyi Hu, Xiaoyuan Yi, Wenhao Li, Maosong Sun, and Xing Xie. 2022. Fuse it more deeply! a variational transformer with layer-wise latent variable inference for text generation. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 697–716, Seattle, United States. Association for Computational Linguistics.

Zhiting Hu and Li Erran Li. 2021. A causal lens for controllable text generation. *Advances in Neural Information Processing Systems*, 34:24941–24955.

Kuan-Hao Huang and Kai-Wei Chang. 2021. Generating syntactically controlled paraphrases without using annotated parallel pairs. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1022–1033, Online. Association for Computational Linguistics.

Peter A Jansen, Elizabeth Wainwright, Steven Marmorstein, and Clayton T Morrison. 2018. Worldtree: A corpus of explanation graphs for elementary science questions supporting multi-hop inference. *arXiv preprint arXiv:1802.03052*.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 424–434, Florence, Italy. Association for Computational Linguistics.

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

Thomas N Kipf and Max Welling. 2016a. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Thomas N Kipf and Max Welling. 2016b. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*.

Sachin Kumar, Eric Malmi, Aliaksei Severyn, and Yulia Tsvetkov. 2021. Controlled text generation as continuous optimization with multiple constraints. In *Advances in Neural Information Processing Systems*.

Bohan Li, Junxian He, Graham Neubig, Taylor Berg-Kirkpatrick, and Yiming Yang. 2019. A surprisingly effective fix for deep latent variable modeling of text. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3603–3614, Hong Kong, China. Association for Computational Linguistics.

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.

Yafu Li, Leyang Cui, Jianhao Yan, Yongjing Yin, Wei Bi, Shuming Shi, and Yue Zhang. 2023. Explicit syntactic guidance for neural text generation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14095–14112, Toronto, Canada. Association for Computational Linguistics.

Zhuang Li, Lizhen Qu, Qiongkai Xu, Tongtong Wu, Tianyang Zhan, and Gholamreza Haffari. 2022. Variational autoencoder with disentanglement priors for low-resource task-specific natural language generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 10335–10356, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Zhun Liu, Ying Shen, Varun Bharadhwaj Lakshminarasimhan, Paul Pu Liang, AmirAli Bagher Zadeh, and Louis-Philippe Morency. 2018. Efficient low-rank multimodal fusion with modality-specific factors. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2247–2256, Melbourne, Australia. Association for Computational Linguistics.

Alireza Makhzani, Jonathon Shlens, Navdeep Jaitly, Ian Goodfellow, and Brendan Frey. 2016. Adversarial autoencoders.

Jordan Meadows, Marco Valentino, Damien Teney, and Andre Freitas. 2023. A symbolic framework for systematic evaluation of mathematical reasoning with transformers. *arXiv preprint arXiv:2305.12563*.

Nathan Michlo, Richard Klein, and Steven James. 2023. Overlooked implications of the reconstruction loss for vae disentanglement.

Melanie Mitchell. 2023. How do we know how smart ai systems are?

Alireza Mohammadshahi and James Henderson. 2023. Syntax-aware graph-to-graph transformer for semantic role labelling. In *Proceedings of the 8th Workshop on Representation Learning for NLP (RepL4NLP 2023)*, pages 174–186, Toronto, Canada. Association for Computational Linguistics.

Khalil Mrini, Emilia Farcas, and Ndapa Nakashole. 2021. Recursive tree-structured self-attention for answer sentence selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4651–4661, Online. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Mark Pinsky and Samuel Karlin. 2010. *An introduction to stochastic modeling*. Academic press.

Amir Pouran Ben Veyseh, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Graph transformer networks with syntactic and semantic structures for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3651–3661, Online. Association for Computational Linguistics.

Lianhui Qin, Vered Shwartz, Peter West, Chandra Bhagavatula, Jena D. Hwang, Ronan Le Bras, Antoine Bosselut, and Yejin Choi. 2020. Back to the future: Unsupervised backprop-based decoding for counterfactual and abductive commonsense reasoning. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 794–805, Online. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Paul K. Rubenstein, Bernhard Schoelkopf, and Ilya Tolstikhin. 2018. On the latent space of wasserstein auto-encoders.

Devendra Sachan, Yuhao Zhang, Peng Qi, and William L. Hamilton. 2021. Do syntax trees help pre-trained transformers extract information? In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2647–2661, Online. Association for Computational Linguistics.

Rajendran Sankaravelayuthan. 2020. Word order typology and language universals.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. Bleurt: Learning robust metrics for text generation.

Tianxiao Shen, Jonas Mueller, Regina Barzilay, and Tommi Jaakkola. 2020. Educating text autoencoders: Latent representation guidance via denoising. In *International Conference on Machine Learning*, pages 8719–8729. PMLR.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 1526–1534.

Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R Bowman, Dipanjan Das, et al. 2019. What do you learn from context? probing for sentence structure in contextualized word representations. *arXiv preprint arXiv:1905.06316*.

Mokanarangan Thayaparan, Marco Valentino, and André Freitas. 2021. Explainable inference over grounding-abstract chains for science questions. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1–12, Online. Association for Computational Linguistics.

Naftali Tishby, Fernando C Pereira, and William Bialek. 2000. The information bottleneck method. *arXiv preprint physics/0004057*.

Marco Valentino, Jordan Meadows, Lan Zhang, and André Freitas. 2023. Multi-operational mathematical derivations in latent space.

Marco Valentino, Mokanarangan Thayaparan, Deborah Ferreira, and André Freitas. 2022. Hybrid autoregressive inference for scalable multi-hop explanation regeneration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11403–11411.

Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(86):2579–2605.

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

Amir Pouran Ben Veyseh, Tuan Ngo Nguyen, and Thien Huu Nguyen. 2020. Graph transformer networks with syntactic and semantic structures for event argument extraction. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3651–3661.

Lingfei Wu, Yu Chen, Kai Shen, Xiaojie Guo, Hanning Gao, Shucheng Li, Jian Pei, Bo Long, et al. 2023. Graph neural networks for natural language processing: A survey. *Foundations and Trends® in Machine Learning*, 16(2):119–328.

Erguang Yang, Mingtong Liu, Deyi Xiong, Yujie Zhang, Yao Meng, Changjian Hu, Jinan Xu, and Yufeng Chen. 2021. Syntactically-informed unsupervised paraphrasing with non-parallel data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2594–2604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Wenlin Yao, Lifeng Jin, Hongming Zhang, Xiaoman Pan, Kaiqiang Song, Dian Yu, Dong Yu, and Jianshu Chen. 2023. How do words contribute to sentence semantics? revisiting sentence embeddings with a perturbation method. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 3001–3010, Dubrovnik, Croatia. Association for Computational Linguistics.

Seongjun Yun, Minbyul Jeong, Raehyun Kim, Jaewoo Kang, and Hyunwoo J. Kim. 2020. Graph transformer networks.

Fabio Massimo Zanzotto, Andrea Santilli, Leonardo Ranaldi, Dario Onorati, Pierfrancesco Tommasino, and Francesca Fallucchi. 2020. KERMIT: Complementing transformer architectures with encoders of explicit syntactic interpretations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 256–267, Online. Association for Computational Linguistics.

Xinyuan Zhang, Yi Yang, Siyang Yuan, Dinghan Shen, and Lawrence Carin. 2019. Syntax-infused variational autoencoder for text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2069–2078, Florence, Italy. Association for Computational Linguistics.

Yingji Zhang, Danilo S Carvalho, Ian Pratt-Hartmann, and André Freitas. 2022. Quasi-symbolic explanatory nli via disentanglement: A geometrical examination. *arXiv preprint arXiv:2210.06230*.

Yingji Zhang, Danilo S Carvalho, Ian Pratt-Hartmann, and André Freitas. 2023a. Learning disentangled semantic spaces of explanations via invertible neural networks. *arXiv preprint arXiv:2305.01713*.

Yingji Zhang, Danilo S Carvalho, Ian Pratt-Hartmann, and Andre Freitas. 2023b. Towards controllable natural language inference through lexical inference types. *arXiv preprint arXiv:2308.03581*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 654–664, Vancouver, Canada. Association for Computational Linguistics.

Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. 2020. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81.

A Training setups

Encoding architecture Figure 6 illustrates the four architectures of encoding baseline for learning syntax representation.

Datasets Table 4 displays the statistical information of the datasets used in the experiment. As for the AutoEncoder setup, we use the non-repetitive explanations selected from both WorldTree (Jansen et al., 2018) and EntailmentBank (Dalvi et al., 2021) corpus as the experimental data. The mathematical expressions are derived from (Meadows et al., 2023).

Corpus	Num data.	Avg. length
WorldTree	11430	8.65
EntailmentBank	5134	10.35
Math Symbol	32000	6.84

Table 4: Statistics from datasets.

Tokenization As for mathematical expression, we add specific math tokens, including $\text{frac}, \sin, \cos, \log, e$, into the dictionary of both Bert and GPT2 and consider the remaining tokens as char-level. As for explanatory sentences, we use the default tokenization in Bert and GPT2.

Syntax parsing As for mathematical expression, we use Expression Trees [4], As for explanatory sentences, we use consistency parser[5] from AllenNLP library (Gardner et al., 2018) to get the flattened syntax tree, and remove all word content from the tree as the input of graph encoder.

Model implementation As for graph encoders, we use *PyTorch Geometric* library [6]. We deployed two hidden layers for GCN, GraphSAGE, and TransformerCONV. For mathematical expression, we replace the content of variables with random noises following uniform distribution with the range between -1 and 1 during the node embedding stage. The implementation of Optimus is based on their original code [7]. The implementation of LSTM-based VAEs is based on the code supplied from Shen et al. (2020) [8].

Hyperparameters In the experiment, all baselines and our architecture hold the same size of latent representation (768). The training epoch is 100, the learning rate is 5e-5, the batch size is 64.

[4] https://docs.sympy.org/latest/tutorials/intro-tutorial/manipulation.html

[5] https://demo.allennlp.org/constituency-parsing

[6] https://pytorch-geometric.readthedocs.io/en/latest/

[7] https://github.com/ChunyuanLI/Optimus

[8] https://github.com/shentianxiao/text-autoencoders

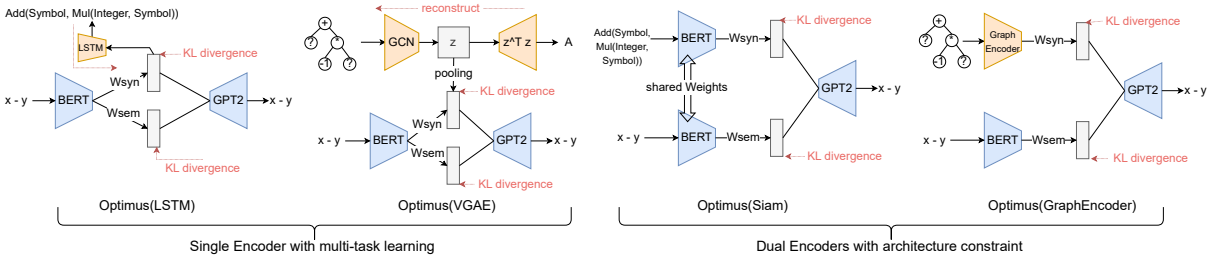

Figure 6: Overview of different methods to explicitly represent and disentangle syntactic information in the latent space of Transformer-based VAEs.

B More Experimental results

Math Semantic visualization Figure 7 visualize the latent space geometry of semantic space.

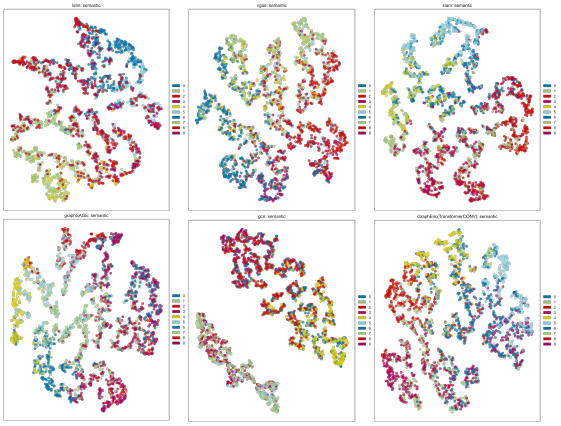

Figure 7: Visualizing semantic space separation (top: LSTM, VGAE, Siam, bottom: graph encoders with GCN, GraphSAGE, TransformerCONV).

Explanations Syntax visualization Figure 8 visualize the latent space geometry of syntax space of explanatory sentences.

Explanations Semantic visualization Figure 9 visualize the latent space geometry of semantic space of explanatory sentences.

Qualitative evaluation Moreover, we randomly sample the points in each k-mean cluster and output the corresponding sentences or syntax parse tree in Table 5, 6, and 7.

Besides, in Table 8, we provide the comparison of reconstructed sentences between normal Optimus and Bert-TransCONV(addition QKV).

Attention heatmap We provide more attention heatmap of different sentences in Figure 10 and 11. Similar observation as before, the latent representation can better capture word content information under the graph-language encoding setup.

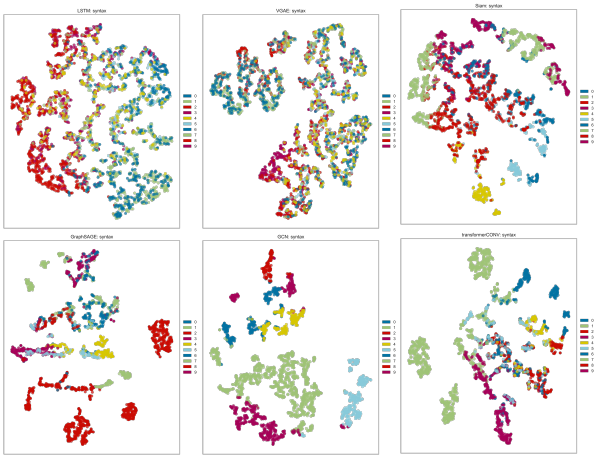

Figure 8: Visualizing syntax space separation (top: LSTM, VGAE, Siam, bottom: graph encoders with GCN, GraphSAGE, TransformerCONV).

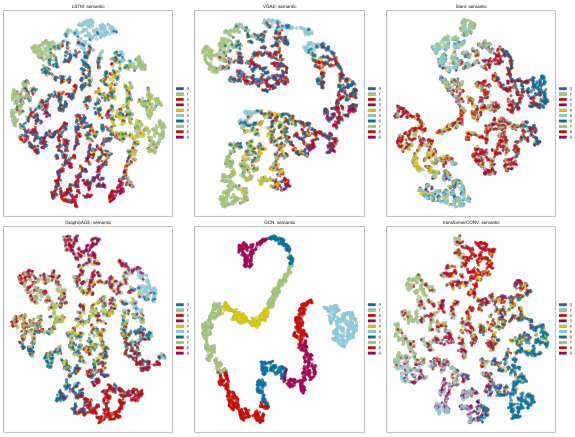

Figure 9: Visualizing semantic space separation (top: LSTM, VGAE, Siam, bottom: graph encoders with GCN, GraphSAGE, TransformerCONV).

Traversal We provide the traversed sentences of semantic space and syntax space in table 9 and 10, respectively. From it, we can observe that the geometrical neighbour sentences traversed via *Ornstein-Uhlenbeck* random walk can hold similar lexical information ("sea/river/ocean").

More specifically, regarding the traversal of the

Math symbol: Syntax Cluster Traversal
C_0: Pow(cos(Symbol(E)), Symbol(b)) C_0: Pow(exp(Symbol(b)), Symbol(A)) C_0: Mul(Symbol(F), sin(Symbol(g))) C_4: exp(Mul(Pow(Symbol(V), Integer(-1)), Symbol(q))) C_4: cos(Mul(Pow(Symbol(b), Integer(-1)), Symbol(g))) C_4: exp(Mul(Pow(Symbol(T), Integer(-1)), Symbol(a))) C_8: sin(Mul(Symbol(A), Symbol(k))) C_8: cos(Mul(Symbol(U), Symbol(w))) C_8: exp(Mul(Symbol(J), Symbol(l)))

Table 5: Qualitative evaluation of syntax cluster of Bert-TransCONV encoding.

Explanations: Semantic Cluster Traversal
C_0: if a pot is exposed to a stove then the pot will become hot C_0: if something is used for something else then that something else is the job of that something C_0: if there is a crack in a rock then water can get into the crack C_8: decaying plant is a source of nutrients in soil C_8: producers are a source of food energy for living things C_8: organic matter is a source of nutrients in soil C_5: a magnet is a kind of object C_5: a board is a kind of object C_5: a wagon is a kind of object

Table 6: Qualitative evaluation of semantic cluster of Bert-GCN encoding.

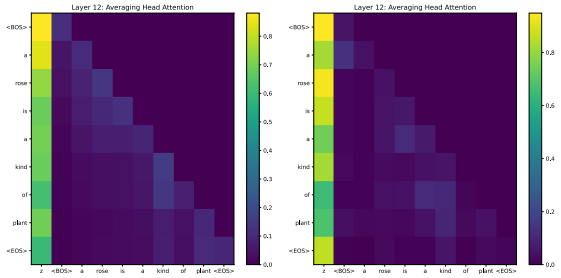

Figure 10: *a rose is a kind of plant*.

syntactic space (Table 10), we can find that the semantics of the generated sentences exhibit higher variability (compared to the variability in syntactic structures when we traverse the semantic space). We conjecture this is mainly because a change in syntactic structure is intrinsically connected with a change in semantics (that is, a perfect separation between the two spaces is extremely hard to

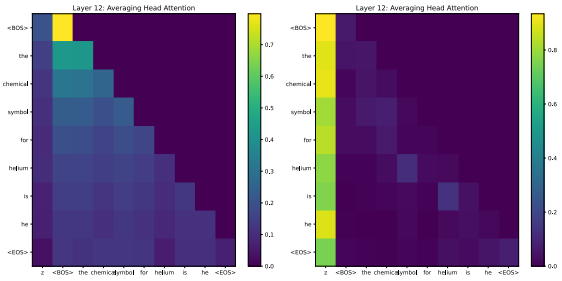

Figure 11: *the chemical symbol for helium is he*.

achieve). For example, the traversal of the syntactic structure such as the one in Table 10 (e.g., from (S (NP) (VP (ADJP (PP (NP))))) —> (S (NP) (VP (NP (NP) (PP (NP (NP) (PP (NP (ADJP(PP (NP)))))))))) will intrinsically require changes in the semantics of the generated sentences. However, while the intrinsic semantics is expected to change, an alleviation of the information bottleneck is expected to reduce at least the lexical variability of the sentences (that is including entities and relations that are more closely related) derived from our semantic-syntactic separation. In this case, we can observe better results when we compare our approach with Optimus.

Explanations: Syntax Cluster Traversal

C_5: (S (NP (JJ) (NN)) (VP (VBZ) (NP (JJ) (NN))))
C_5: (S (NP (DT) (NN)) (VP (VBZ) (NP (DT) (NN))))
C_5: (S (NP (JJ) (JJ) (NN)) (VP (VBZ) (NP (JJ) (NN))))

C_6: (S (NP (NN)) (VP (VBZ) (PP (IN) (NP (NP (DT) (NN)) (SBAR (WHNP (WDT)) (S (VP (VBZ) (VP (VBN) (PP (IN) (NP (NN))))))))))
C_6: (S (NP (NN)) (VP (VBZ) (NP (NP (DT) (NN)) (PP (IN) (SBAR (WHADVP (WRB)) (S (NP (DT) (NN)) (VP (VBZ) (VP (VBN))))))))))
C_6: (S (NP (NN)) (VP (VBZ) (NP (NP (DT) (NN)) (SBAR (WHNP (WDT)) (S (VP (VBZ) (ADJP (JJ) (JJS) (PP (IN) (NP (DT) (NNP))))))))))

C_9: (S (NP (NNS)) (VP (VBP) (NP (NN)) (PP (IN) (NP (NNS)))))
C_9: (S (NP (NNS)) (VP (VBP) (PP (IN) (NP (NN)))))
C_9: (S (NP (NNS)) (VP (MD) (VP (VB) (NP (NN) (NN)) (PP (IN) (NP (DT) (NN))))))

Table 7: Qualitative evaluation of semantic cluster of Bert-GCN encoding.

Gold explanations	**BERT-GPT2**	**Bert/TransCONV-GPT2**
lenses are a kind of object	frog is a kind of object	lenses are a kind of object
the chemical symbol for helium is he	a substance has a physical shape	the chemical symbol for helium is He
a rose is a kind of plant	a window pane is a kind of surface	a rose is a kind of flower
a body of water contains water	a flood has a large amount of rainfall	a body of water contains water
growing is a kind of process	population is a kind of process	growing is a kind of process
air is a kind of gas	farming is a kind of human	air is a kind of gas
action means activity	feed means use	activity means action
soda water is a kind of carbonated beverage	condensing is a kind of change in temperature	soda water is a kind of carbonated beverage
plasma is a kind of state of matter	black probability is a kind of event	plasma is a kind of state of matter
earth is a kind of celestial object	sun is a kind of light	earth is a kind of celestial object
a bee is a kind of living thing	a frog is a kind of amphibian	a bee is a kind of living thing
green is a kind of color	deforestation is a kind of process	green is a kind of color
a wooded area is a kind of forest	a coal mine is a kind of natural resource	a wooded area is a kind of forest

Table 8: Explanation reconstruction (left: original explanations from WorldTree corpus, middle: explanations from Optimus, right: explanations from Bert-TransCONV (addition Q)).

Semantic Space Traversal

Optimus:
0: a desert is a land found in desert environments
1: a forest is a large structure that contains lots of trees
2: a river is a nonliving thing
3: a canyon is a very deep valley
4: a mountain is a large land mass

Bert-TransCONV:
0: a sea is a source of water for humans
1: a sea is a source of freshwater
2: a river is a source of water
3: an ocean is a source of water for residents

Table 9: Qualitative evaluation of traversed examples of Optimus (top) and Bert-TransCONV (addition QKV) (bottom).

Syntax Space Traversal

Bert-TransCONV:
0: a river is synonymous with a coastline
1: a hurricane is composed of water vapor and dust
2: a hurricane is the source of most of water vapor in the atmosphere
3: hurricane is mainly made of water vapor
4: a hurricane is measuring the amount of water in an area

Table 10: Qualitative evaluation of traversed examples of Bert-TransCONV (addition QKV).