Hierarchical Sketch Induction for Paraphrase Generation

Tom Hosking   **Hao Tang**   **Mirella Lapata**

Institute for Language, Cognition and Computation
School of Informatics, University of Edinburgh
10 Crichton Street, Edinburgh EH8 9AB
tom.hosking@ed.ac.uk   hao.tang@ed.ac.uk   mlap@inf.ed.ac.uk

Abstract

We propose a generative model of paraphrase generation, that encourages syntactic diversity by conditioning on an explicit syntactic sketch. We introduce Hierarchical Refinement Quantized Variational Autoencoders (HRQ-VAE), a method for learning decompositions of dense encodings as a sequence of discrete latent variables that make iterative refinements of increasing granularity. This hierarchy of codes is learned through end-to-end training, and represents fine-to-coarse grained information about the input. We use HRQ-VAE to encode the syntactic form of an input sentence as a path through the hierarchy, allowing us to more easily predict syntactic sketches at test time. Extensive experiments, including a human evaluation, confirm that HRQ-VAE learns a hierarchical representation of the input space, and generates paraphrases of higher quality than previous systems.

1 Introduction

Humans use natural language to convey information, mapping an abstract idea to a sentence with a specific surface form. A paraphrase is an alternative surface form of the same underlying semantic content. The ability to automatically identify and generate paraphrases is of significant interest, with applications in data augmentation (Iyyer et al., 2018), query rewriting, (Dong et al., 2017) and duplicate question detection (Shah et al., 2018).

While autoregressive models of language (including paraphrasing systems) predict one token at a time, there is evidence that in humans some degree of planning occurs at a higher level than individual words (Levelt, 1993; Martin et al., 2010). Prior work on paraphrase generation has attempted to include this inductive bias by specifying an alternative surface form as additional model input, either in the form of target parse trees (Iyyer et al., 2018; Chen et al., 2019a; Kumar et al., 2020), exemplars (Meng et al., 2021), or syntactic codes
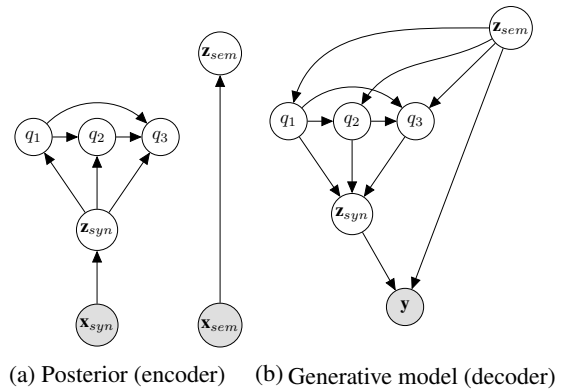

(a) Posterior (encoder)   (b) Generative model (decoder)

Figure 1: The generative models underlying our approach. Given some semantic content $\mathbf{z}_{sem}$, we predict a hierarchical set of syntactic codes q_d that describe the output syntactic form at increasing levels granularity. These are combined to give a syntactic embedding $\mathbf{z}_{syn}$, which is fed to the decoder along with the original semantic content to generate the output sentence $\mathbf{y}$. During training, the encoder is driven by a paraphrase $\mathbf{x}_{sem}$ and a syntactic exemplar $\mathbf{x}_{syn}$.

(Shu et al., 2019; Hosking and Lapata, 2021). Most of these approaches suffer from an 'all or nothing' problem: the target surface form must be fully specified during inference. However, predicting the complete syntactic structure is almost as difficult as predicting the sentence itself, negating the benefit of the additional planning step.

In this paper, we propose a generative model for paraphrase generation, that combines the diversity introduced by an explicit syntactic target with the tractability of models trained end-to-end. Shown in Figure 1, the model begins by assuming the existence of some semantic content $\mathbf{z}_{sem}$. Conditioned on this semantic information, the model predicts a syntactic 'sketch' in the form of a hierarchical set of discrete codes $q_{1:D}$, that describe the target syntactic structure with increasing granularity. The sketch is combined into an embedding $\mathbf{z}_{syn}$, and fed along with the original meaning $\mathbf{z}_{sem}$ to a de-

coder that generates the final output utterance **y**. Choosing a discrete representation for the sketch means it can be predicted from the meaning as a simple classification task, and the hierarchical nature means that the joint probability over the codes admits an autoregressive factorisation, making prediction more tractable.

The separation between $\mathbf{z}_{sem}$ and $\mathbf{z}_{syn}$ is induced by a training scheme introduced in earlier work (Hosking and Lapata, 2021; Huang and Chang, 2021) and inspired by prior work on separated latent spaces (Chen et al., 2019b; Bao et al., 2019), whereby the model must reconstruct a target output from one input with the correct meaning, and another input with the correct syntactic form. To learn the discretized sketches, we propose a variant of Vector-Quantized Variational Autoencoders (VQ-VAE, or VQ) that learns a *hierarchy of embeddings* within a shared vector space, and represents an input encoding as a path through this hierarchy. Our approach, which we call Hierarchical Refinement Quantized Variational Autoencoders or **HRQ-VAE**, leads to a decomposition of a dense vector into embeddings of increasing granularity, representing high-level information at the top level before gradually refining the encoding over subsequent levels.

Our contributions are summarized as follows:

- We propose a generative model of natural language generation, HRQ-VAE, that induces a syntactic sketch to account for the diversity exhibited by paraphrases. We present a parameterization of our generative model that is a novel method for learning hierarchical discretized embeddings over a single latent encoding space. These embeddings are trained end-to-end and jointly with the encoder/decoder.

- We use HRQ-VAE to induce hierarchical sketches for paraphrase generation, demonstrating that the known factorization over codes makes them easier to predict at test time, and leads to higher quality paraphrases.

2 Latent Syntactic Sketches

2.1 Motivation

Let **y** be a sentence, represented as a sequence of tokens. We assume that **y** contains semantic content, that can be represented by a latent variable $\mathbf{z}_{sem}$. Types of semantic content might include the description of an image, or a question intent. How-

ever, the mapping from semantics to surface form is not unique: in general, there is more than one way to express the semantic content. Sentences with the same underlying meaning $\mathbf{z}_{sem}$ but different surface form **y** are *paraphrases*. Standard approaches to paraphrasing (e.g., Bowman et al. 2016) map directly from $\mathbf{z}_{sem}$ to **y**, and do not account for this diversity of syntactic structure.

Following recent work on syntax-guided paraphrasing (Chen et al., 2019a; Hosking and Lapata, 2021), and inspired by evidence that humans plan out utterances at a higher level than individual words (Martin et al., 2010), we introduce an intermediary *sketching* step, depicted in Figure 1b. We assume that the output sentence **y** is generated as a function both of the meaning $\mathbf{z}_{sem}$ *and* of a syntactic encoding $\mathbf{z}_{syn}$ that describes the structure of the output. Moreover, since natural language displays hierarchical organization in a wide range of ways, including at a syntactic level (constituents may contain other consituents), we also assume that the syntactic encoding $\mathbf{z}_{syn}$ can be decomposed into a hierarchical set of discrete latent variables $q_{1:D}$, and that these q_d are conditioned on the meaning $\mathbf{z}_{sem}$. This contrasts with popular model architectures such as VAE (Bowman et al., 2015) which use a *flat* internal representation in a dense Euclidean vector space.

Intuitively, our generative model corresponds to a process where a person thinks of a message they wish to convey; then, they decide roughly how to say it, and incrementally refine this decision; finally, they combine the meaning with the syntactic sketch to 'spell out' the sequence of words making up the sentence.

2.2 Factorization and Objective

The graphical model in Figure 1b factorizes as

$$
\begin{aligned}
p(\mathbf{y}, \mathbf{z}_{sem}) = \sum_{q_{1:D}, \mathbf{z}_{syn}} & p(\mathbf{y}|\mathbf{z}_{sem}, \mathbf{z}_{syn}) \\
& \times p(\mathbf{z}_{syn}|q_{1:D})
\end{aligned}
$$
$$
\times p(\mathbf{z}_{sem}) \times p(q_1|\mathbf{z}_{sem}) \prod_{d=2}^{D} \times p(q_d|q_{<d}, \mathbf{z}_{sem}). \quad (1)
$$

Although $q_{1:D}$ are conditionally dependent on $\mathbf{z}_{sem}$, we assume that $\mathbf{z}_{sem}$ may be determined from **y** without needing to explicitly calculate $q_{1:D}$ or $\mathbf{z}_{syn}$. We also assume that the mapping from discrete codes $q_{1:D}$ to $\mathbf{z}_{syn}$ is a deterministic func-

tion $f_{q \to \mathbf{z}}(\cdot)$. The posterior therefore factorises as

$$\phi(\mathbf{z}_{sem}, \mathbf{z}_{syn}|\mathbf{y}) = \phi(\mathbf{z}_{sem}|\mathbf{y}) \times \phi(\mathbf{z}_{syn}|\mathbf{y})$$
$$\times \phi(q_1|\mathbf{z}_{syn}) \times \prod_{d=2}^{D} \phi(q_d|q_{<d}, \mathbf{z}_{syn}). \quad (2)$$

The separation between $\mathbf{z}_{sem}$ and $q_{1:D}$, such that they represent the meaning and form of the input respectively, is induced by the training scheme. During training, the model is trained to reconstruct a target $\mathbf{y}$ using $\mathbf{z}_{sem}$ derived from an input with the correct meaning (a paraphrase) $\mathbf{x}_{sem}$, and $q_{1:D}$ from another input with the correct form (a syntactic exemplar) $\mathbf{x}_{syn}$. Hosking and Lapata (2021) showed that the model therefore learns to encode primarily semantic information about the input in $\mathbf{z}_{sem}$, and primarily syntactic information in $q_{1:D}$. Exemplars are retrieved from the training data following to the process described in Hosking and Lapata (2021), with examples in Appendix C. The setup is shown in Figure 1a; in summary, during training we set $\phi(\mathbf{z}_{sem}|\mathbf{y}) = \phi(\mathbf{z}_{sem}|\mathbf{x}_{sem})$ and $\phi(q_d|\mathbf{y}, q_{<d}) = \phi(q_d|\mathbf{x}_{syn}, q_{<d})$. The final objective is given by

$$\text{ELBO} = \mathbb{E}_{\phi}\big[-\log p(\mathbf{y}|\mathbf{z}_{sem}, q_{1:D}))$$
$$-\log p(q_1|\mathbf{z}_{sem}) - \sum_{d=2}^{D} \log p(q_d|q_{<d}, \mathbf{z}_{sem})\big]$$
$$+ KL\big[\phi(\mathbf{z}_{sem}|\mathbf{x}_{sem})||p(\mathbf{z}_{sem})\big], \quad (3)$$

where $q_d \sim \phi(q_d|\mathbf{x}_{syn})$ and $\mathbf{z}_{sem} \sim \phi(\mathbf{z}_{sem}|\mathbf{x}_{sem})$.

3 Neural Parameterisation

We assume a Gaussian distribution for $\mathbf{z}_{sem}$, with prior $p(\mathbf{z}_{sem}) \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. The encoders $\phi(\mathbf{z}_{sem}|\mathbf{x}_{sem})$ and $\phi(\mathbf{z}_{syn}|\mathbf{x}_{syn})$ are Transformers (Vaswani et al., 2017), and we use an autoregressive Transformer decoder for $p(\mathbf{y}|\mathbf{z}_{sem}, \mathbf{z}_{syn})$. The mapping $f_{q \to \mathbf{z}}(\cdot)$ from $q_{1:D}$ to $\mathbf{z}_{syn}$ and the posterior network $\phi(q_d|q_{<d}, \mathbf{z}_{syn})$ are more complex, and form a significant part of our contribution.

Our choice of parameterization is learned end-to-end, and ensures that the sketches learned are hierarchical both in the shared embedding space and in the information they represent.

3.1 Hierarchical Refinement Quantization

Let $\mathbf{z}_{syn} \in \mathcal{R}^{\mathbb{D}}$ be the output of the encoder network $\phi(\mathbf{z}_{syn}|\mathbf{y})$, that we wish to decompose as a sequence of discrete hierarchical codes. Recall that $q_d \in [1, K]$ are discrete latent variables corresponding to the codes at different levels in the hierarchy, $d \in [1, D]$. Each level uses a distinct codebook, $\mathbf{C}_d \in \mathbb{R}^{K \times \mathbb{D}}$, which maps each discrete code to a continuous embedding $\mathbf{C}_d(q_d) \in \mathbb{R}^{\mathbb{D}}$.

The distribution over codes at each level is a softmax distribution, with the scores s_d given by the distance from each of the codebook embeddings to the residual error between the input and the cumulative embedding from all previous levels,

$$s_d(q) = -\left(\left[\mathbf{x} - \sum_{d'=1}^{d-1} \mathbf{C}_{d'}(q_{d'})\right] - \mathbf{C}_d(q)\right)^2. \quad (4)$$

Illustrated in Figure 2, these embeddings therefore represent iterative refinements on the quantization of the input. The posterior network $\phi(q_d|q_{<d}, \mathbf{z}_{syn})$ iteratively *decomposes* an encoding vector into a path through a hierarchy of clusters whose centroids are the codebook embeddings.

Given a sequence of discrete codes $q_{1:D}$, we deterministically construct its continuous representation with the composition function $f_{q \to \mathbf{z}}(\cdot)$,

$$\mathbf{z}_{syn} = f_{q \to \mathbf{z}}(q_{1:D}) = \sum_{d=1}^{D} \mathbf{C}_d(q_d). \quad (5)$$

HRQ-VAE can be viewed as an extension of VQ-VAE (van den Oord et al., 2017), with two significant differences: (1) the codes are hierarchically ordered and the joint distribution $p(q_1, \ldots, q_D)$ admits an autoregressive factorization; and (2) the HRQ-VAE composition function is a sum, compared to concatenation in VQ or a complex neural network in VQ-VAE 2 (Razavi et al., 2019). Under HRQ, latent codes describe a path through the learned hierarchy within a shared encoding space. The form of the posterior $\phi(q_d|q_{<d}, \mathbf{z}_{syn})$ and the composition function $f_{q \to \mathbf{z}}(\cdot)$ do not rely on any particular properties of the paraphrasing task; the technique could be applied to any encoding space.

Initialisation Decay Smaller perturbations in encoding space should result in more fine grained changes in the information they encode. Therefore, we encourage *ordering* between the levels of hierarchy (such that lower levels encode finer grained information) by initialising the codebook with a decaying scale, such that later embeddings have a smaller norm than those higher in the hierarchy. Specifically, the norm of the embeddings at level d is weighted by a factor $(\alpha_{init})^{d-1}$.
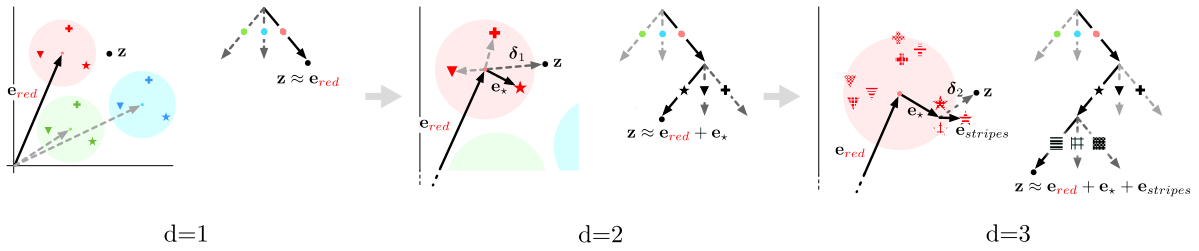
d=1   d=2   d=3

Figure 2: An illustration of how HRQ-VAE maps an input encoding vector $\mathbf{z}$ to a decomposition of hierarchical discretized encodings. HRQ-VAE compares the input to a jointly learned codebook of embeddings that become increasingly granular at lower depths of hierarchy. In this simplified example, with a depth of 3 and a codebook size of 3, the nearest top-level (colours) embedding to $\mathbf{z}$ is $\mathbf{e}_{red}$; then, the residual error $\boldsymbol{\delta}_1 = \mathbf{z} - \mathbf{e}_{red}$ is compared to the 2nd level of embeddings (shapes), with the nearest being $\mathbf{e}_\star$. Finally, the residual error $\boldsymbol{\delta}_2$ is compared to the 3rd level codebook (patterns), where the closest is $\mathbf{e}_{stripes}$. The quantized encoding of $\mathbf{z}$ is then $\mathbf{z} \approx \mathbf{e}_{red} + \mathbf{e}_\star + \mathbf{e}_{stripes}$.

Depth Dropout To encourage the hierarchy within the encoding space to correspond to hierarchical properties of the output, we introduce *depth dropout*, whereby the hierarchy is truncated at each level during training with some probability p_{depth}. The output of the quantizer is then given by

$$\mathbf{z}_{syn} = \sum_{d=1}^{D} \left(\mathbf{C}_d(q_d) \prod_{d'=1}^{d} \gamma_{d'} \right), \quad (6)$$

where $\gamma_h \sim \text{Bernoulli}(1 - p_{depth})$. This means that the model is sometimes trained to reconstruct the output based only on a *partial* encoding of the input, and should learn to cluster similar outputs together at each level in the hierarchy.

3.2 Sketch Prediction Network

During training the decoder is driven using sketches sampled from the encoder, but at test time exemplars are unavailable and we must predict a distribution over syntactic sketches $p(q_{1:D}|\mathbf{z}_{sem})$. Modelling the sketches as hierarchical ensures that this distribution admits an autoregressive factorization.

We use a simple recurrent network to infer valid codes at each level of hierarchy, using the semantics of the input sentence and the cumulative embedding of the predicted path so far as input, such that q_d is sampled from $p(q_d|\mathbf{z}_{sem}, q_{<d}) = \text{Softmax}(\text{MLP}_d(\mathbf{z}_{sem}, \mathbf{z}_{<d}))$, where $\mathbf{z}_{<d} = \sum_{d'=1}^{d-1} \mathbf{C}_{d'}(q_{d'})$. This MLP is trained jointly with the encoder/decoder model, using the outputs of the posterior network $\phi(q_d|\mathbf{x}_{syn}, q_{<d})$ as targets. To generate paraphrases as test time, we sample from the sketch prediction model $p(q_d|\mathbf{z}_{sem}, q_{<d})$ using beam search and condition generation on these predicted sketches.

3.3 Training Setup

We use the Gumbel reparameterisation trick (Jang et al., 2016; Maddison et al., 2017; Sønderby et al., 2017) for the discrete codes and the standard Gaussian reparameterisation for the semantic representation. To encourage the model to use the full codebook, we decayed the Gumbel temperature τ, according to the schedule given in Appendix A. We approximate the expectation in Equation (3) by sampling from the training set and updating via backpropagation (Kingma and Welling, 2014). The full model was trained jointly by optimizing the ELBO in Equation (3).

4 Experimental Setup

Datasets A paraphrase is 'an alternative surface form in the same language expressing the same semantic content as the original form' (Madnani and Dorr, 2010), but it is not always clear what counts as the 'same semantic content'. Our approach requires access to reference paraphrases; we evaluate on three English paraphrasing datasets which have clear grounding for the meaning of each sentence: Paralex (Fader et al., 2013), a dataset of question paraphrase clusters scraped from WikiAnswers; Quora Question Pairs (QQP)[1] sourced from the community question answering forum Quora; and MSCOCO 2017 (Lin et al., 2014), a set of images that have been captioned by multiple annotators. For the question datasets, each paraphrase is grounded to the (hypothetical) *answer* they share. We use the splits released by Hosking and Lapata (2021). For MSCOCO, each caption is grounded by the *image* that it describes. We evaluate on the public validation set, randomly selecting one cap-
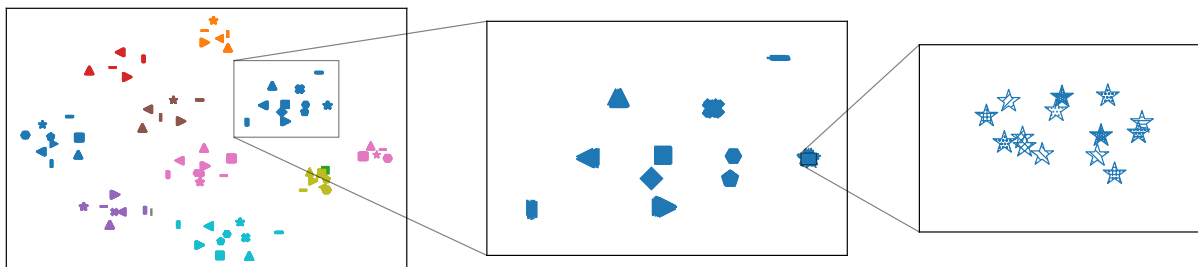
---

[1] https://www.kaggle.com/c/quora-question-pairs

Figure 3: t-SNE visualisation of the syntactic encodings $\mathbf{z}_{syn}$ for 10k examples from Paralex: colours indicate top-level codes q_1, shapes indicate the second level, and patterns are used to label the third level. Deeper levels in the hierarchy represent finer grained information in encoding space.

tion for each image to use as input and using the remaining four as references.

Model Configuration Hyperparameters were tuned on the Paralex development set, and reused for the other evaluations. We set the depth of the hierarchy $D = 3$, and the codebook size $K = 16$. The Transformer encoder and decoder consist of 5 layers each, and we use the vocabulary and token embeddings from BERT-Base (Devlin et al., 2018). We use an initialisation decay factor of $\alpha_{init} = 0.5$, and a depth dropout probability $p_{depth} = 0.3$. A full set of hyperparameters is given in Appendix A, and our code is available at https://github.com/tomhosking/hrq-vae.

Comparison Systems As baselines, we consider three popular architectures: a vanilla autoencoder (AE) that learns a single dense vector representation of an input sentence; a Gaussian Variational AutoEncoder (VAE, Bowman et al., 2015), which learns a distribution over dense vectors; and a Vector-Quantized Variational AutoEncoder (VQ-VAE, van den Oord et al., 2017), that represents the full input sentence as a set of discrete codes. All three models are trained to generate a sentence from one of its paraphrases in the training data, and are not trained with an autoencoder objective. We implement a simple tf-idf baseline (Jones, 1972), retrieving the question from the training set with the highest cosine similarity to the input. Finally, we include a basic copy baseline as a lower bound, that simply uses the input sentences as the output.

We also compare to a range of recent paraphrasing systems. Latent bag-of-words (BoW, Fu et al., 2019) uses an encoder-decoder model with a discrete bag-of-words as the latent encoding. SOW/REAP (Goyal and Durrett, 2020) uses a two stage approach, deriving a set of feasible syntactic rearrangements that is used to guide a second encoder-decoder model. BTmPG (Lin and Wan,

2021) uses multi-round generation to improve diversity and a reverse paraphrasing model to preserve semantic fidelity. We use the results after 10 rounds of paraphrasing. Separator (Hosking and Lapata, 2021) uses separated, non-hierarchical encoding spaces for the meaning and form of an input, and an additional inference model to predict the target syntactic form at test time. All comparison systems were trained and evaluated on our splits of the datasets.

As an upper bound, we select a sentence from the evaluation set to use as an *oracle* syntactic exemplar, conditioning generation on a sketch that is known to represent a valid surface form.

5 Results

Our experiments were designed to test two primary hypotheses: (1) Does HRQ-VAE learn *hierarchical* decompositions of an encoding space? and (2) Does our choice of generative model enable us to generate *high quality* and *diverse* paraphrases?

5.1 Probing the Hierarchy

Figure 3 shows a t-SNE (van der Maaten and Hinton, 2008) plot of the syntactic encodings $\mathbf{z}_{syn}$ for 10,000 examples from Paralex. The encodings are labelled by their quantization, so that colours indicate top-level codes q_1, shapes denote q_2, and patterns q_3. The first plot shows clear high level structure, with increasingly fine levels of substructure visible as we zoom into each cluster. This confirms that the discrete codes are ordered, with lower levels in the hierarchy encoding more fine grained information.

To confirm that intermediate levels of hierarchy represent valid points in the encoding space, we generate paraphrases using oracle sketches, but truncate the sketches at different depths. Masking one level (i.e., using only q_1, q_2) reduces performance by 2.5 iBLEU points, and two levels by 5.5.

System	Paralex			QQP			MSCOCO		
	BLEU ↑	Self-B ↓	iBLEU ↑	BLEU ↑	Self-B ↓	iBLEU ↑	BLEU ↑	Self-B ↓	iBLEU ↑
Copy	37.10	100.00	9.68	34.52	100.00	7.61	19.85	100.00	-4.12
tf-idf	25.08	25.25	15.01	24.05	62.49	6.75	18.26	38.37	6.93
AE	40.10	75.71	16.94	28.99	60.11	11.17	27.90	38.71	14.58
VAE	38.91	53.28	20.47	27.23	51.09	11.57	27.44	24.40	16.99
VQ-VAE	40.26	65.71	19.07	16.31	21.13	8.83	25.62	22.41	16.01
SOW/REAP	33.09	37.07	19.06	21.27	38.01	9.41	12.51	6.47	8.71
LBoW	34.96	35.86	20.80	23.51	42.08	10.39	21.65	16.46	14.02
BTmPG	28.40	35.99	15.52	19.83	35.11	8.84	19.76	13.04	13.20
Separator	36.36	35.37	22.01	23.68	24.20	14.10	20.59	12.76	13.92
HRQ-VAE	39.49	33.30	**24.93**	33.11	40.35	**18.42**	27.90	16.58	**19.04**
Oracle	50.58	28.09	34.85	50.47	36.84	33.01	35.80	12.85	26.07

Table 1: Top-1 paraphrase generation results, without access to oracle sketches. HRQ-VAE achieves the highest iBLEU scores, indicating the best tradeoff between quality and diversity. Paired bootstrap resampling (Koehn, 2004) indicates that HRQ-VAE significantly improves on all other systems (p< 0.05).

(iBLEU is an automatic metric for assessing paraphrase quality; see Section 5.2). Although encodings using the full depth are the most informative, *partial* encodings still lead to good quality output, with a gradual degradation. This implies both that each level in the hierarchy contains useful information, and that the cluster centroids at each level are representative of the individual members of those clusters.

5.2 Paraphrase Generation

Metrics   Our primary metric is iBLEU (Sun and Zhou, 2012),

$$\text{iBLEU} = \alpha\text{BLEU}(outputs, references) \\ -(1-\alpha)\text{BLEU}(outputs, inputs), \quad (7)$$

that measures the fidelity of generated outputs to reference paraphrases as well as the level of diversity introduced. We use the corpus-level variant. Following the recommendations of Sun and Zhou (2012), we set $\alpha = 0.8$, with a sensitivity analysis shown in Appendix A. We also report $\text{BLEU}(outputs, references)$ as well as $\text{Self-BLEU}(outputs, inputs)$. The latter allows us to examine the extent to which models generate paraphrases that differ from the original input.

To evaluate the diversity between multiple candidates generated by the *same system*, we report pairwise-BLEU (Cao and Wan, 2020),

$$\text{P-BLEU} = \mathbb{E}_{i \neq j}[\text{BLEU}(outputs_i, outputs_j)].$$

This measures the average similarity between the different candidates, with a lower score indicating more diverse hypotheses.

Paralex	Where is the birthplace of woman pro golfer Dottie Pepper?
VAE	Where is the birthplace of Pepper pro golfer Dottie?
BTmpG	What is the birthplace of women pro golfer?
SOW/REAP	What is the birthplace for golfer?
Latent BoW	Where did the golfer golfer originate?
Separator	Where is the birthplace of Dottie?
HRQ-VAE	Where is Dottie Pepper from?
QQP	What are the best ways to defrost lobster tails?
VAE	What are the best ways to defrost lobster tails?
BTmpG	How can I defrost my tails??
SOW/REAP	What is defrost?
Latent BoW	How do you something a something lobster?
Separator	What are some of the best ways to defrost chicken?
HRQ-VAE	How do you thaw frozen lobster tails?
MSCOCO	Set of toy animals sitting in front of a red wooden wagon.
VAE	Two stuffed animals sitting in front of a toy train.
BTmpG	A herd of sheep grazing in a field of grass.
SOW/REAP	A close up of a close up of a street
Latent BoW	A toy wagon with a toy horse and a toy wagon.
Separator	A toy model of a toy horse and buggy.
HRQ-VAE	A group of stuffed animals sitting next to a wooden cart.

Table 2: Examples of generated paraphrases. HRQ-VAE is able to preserve the original meaning, while introducing significant syntactic variation.

Automatic Evaluation   Shown in Table 1, the results of the automatic evaluation highlight the importance of measuring both paraphrase quality and similarity to the input: the Copy baseline is able to achieve high BLEU scores despite simply duplicating the input. The VAE baseline is competitive but tends to have a hi

gh Self-BLEU score, indicating that the semantic preservation comes at the cost of low syntactic diversity. HRQ-VAE achieves both higher BLEU scores and higher iBLEU scores than the comparison systems, indicating that it is able to generate

q_1	q_2	q_3	Output
	Input		Two types of fats in body ?
0	3	6	What types of fats are in a body?
	13	7	What types of fats are there in body?
2	1	2	How many types of fats are there in the body?
	3	7	How many types of fats are there in a body?
5	3	6	What are the different types of fats in a body?
	5	7	What are the different types of fats in a body?
	8	7	Types of fats are different from body fat?
		14	Two types of fats in body?
13	0	2	What are the different types of fats in the body?
		6	What are the different types of fats in a body?
	3	7	What are two types of fats in a body?
		7	What are the different types of fats in body?
	5	8	What are the different types of fats?
		14	What are the different types of fats in the body?

Table 3: Examples of model output, for a range of different sketches. The left hand side shows the sketch (i.e., the values of the codes $q_{1:D}$), with the corresponding model output on the right. q_1 primarily specifies the wh- word (e.g., outputs with $q_1 = 13$ are all 'what' questions), while q_2, q_3 correspond to more fine grained details, e.g., the outputs with $q_3 = 6$ all use the article 'a' when referring to 'body'.

Input	Two types of fat in body?
Exemplar	How many states are in the USA?
No sketch	What are the different types of fats in the body?
q_1	How many types of fats are there in the body?
q_1, q_2	How many fats does the body have?
q_1, q_2, q_3	How many fat are in the body?

Table 4: Model output for varying sketch granularities. When no sketch is used, the model defaults to the most common phrasing of the question. As more detail is included, the output converges towards the exemplar.

Model	Paralex		QQP		MSCOCO	
	iBLEU↑	P-BLEU↓	iBLEU↑	P-BLEU↓	iBLEU↑	P-BLEU↓
VAE	20.49	67.62	11.52	64.71	17.22	55.66
BTmPG	15.50	89.20	9.13	82.02	13.20	80.38
Separator	21.67	62.98	13.63	**52.87**	13.77	57.79
HRQ-VAE	**22.75**	**40.48**	**17.49**	57.29	**18.39**	**41.29**

Table 5: Top-3 generation results. P-BLEU indicates the similarity between the different candidates, while iBLEU scores reported are the mean across the 3 candidates. HRQ-VAE is able to generate multiple high quality paraphrases with more diversity between them than comparison systems.

higher quality paraphrases without compromising on syntactic diversity.

The examples in Table 2 demonstrate that HRQ is able to introduce significant syntactic variation while preserving the original meaning of the input. However, there is still a gap between generation using predicted sketches and 'oracle' sketches (i.e., when the target syntactic form is known in advance), indicating ample scope for improvement.

Worked Example Since the sketches $q_{1:D}$ are latent variables, interpretation is difficult. However, a detailed inspection of example output reveals some structure.

Table 3 shows the model output for a single semantic input drawn from Paralex, across a range of different syntactic sketches. It shows that q_1 is primarily responsible for encoding the question type, with $q_1 = 13$ leading to 'what' questions and $q_1 = 2$ 'how' questions. q_2 and q_3 encode more fine grained details; for example, all outputs shown with $q_3 = 6$ use the indefinite article 'a'.

We also examine how using increasingly granular sketches refines the syntactic template of the output. Table 4 shows the model output for a single semantic input, using varying granularities of sketch extracted from the exemplar. When no sketch is specified, the model defaults to a canonical phrasing of the question. When only q_1 is specified, the output becomes a 'how many' question, and

when a full sketch is included, the output closely resembles the exemplar.

Generating Multiple Paraphrases We evaluated the ability of our system to generate multiple diverse paraphrases for a single input, and compared to the other comparison systems capable of producing more than one output. For both HRQ-VAE and Separator, we used beam search to sample from the sketch prediction network as in the top-1 case, and condition generation on the top-3 hypotheses predicted. For BTmPG, we used the paraphrases generated after 3, 6 and 10 rounds. For the VAE, we conditioned generation on 3 different samples from the encoding space. The results in Table 5 show that HRQ-VAE is able to generate multiple high quality paraphrases for a single input, with lower similarity between the candidates than other systems.

5.3 Human Evaluation

In addition to automatic evaluation we elicited judgements from crowdworkers on Amazon Mechanical Turk. They were shown a sentence and two paraphrases, each generated by a different system, and asked to select which one was preferred along three dimensions: the *dissimilarity* of the paraphrase compared to the original sentence; how
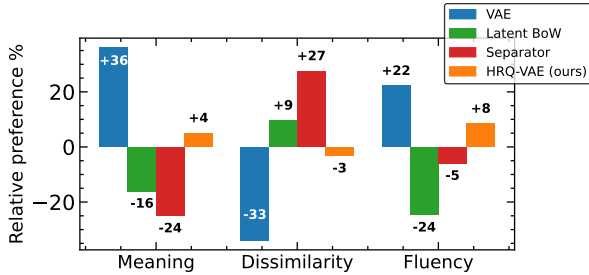
Figure 4: Results of our human evaluation. Although the VAE baseline is the best at preserving sentence meaning, it is the worst at introducing variation to the output. HRQ-VAE offers the best balance between dissimilarity and meaning preservation, and is more fluent than both Separator and Latent BoW.

Variant	Paralex	QQP	MSCOCO
HRQ-VAE (oracle)	34.85	33.01	26.07
No initialisation scaling	−3.06	−2.48	−3.02
No hierarchy	−8.84	−12.72	−3.10
HRQ-VAE	24.93	18.42	19.04
No head dropout	−0.62	−0.74	−0.81
Post-hoc k-means	−3.30	−5.35	−2.83

Table 6: Changes in iBLEU score for a range of ablations from our full model. All components lead to an improvement in paraphrase quality across datasets.

well the paraphrase reflected the *meaning* of the original; and the *fluency* of the paraphrase (see Appendix B). We evaluated a total of 300 sentences sampled equally from each of the three evaluation datasets, and collected 3 ratings for each sample. We assigned each system a score of $+1$ when it was selected, -1 when the other system was selected, and took the mean over all samples. Negative scores indicate that a system was selected less often than an alternative. We chose the four best performing models for our evaluation: HRQ-VAE, Separator, Latent BoW, and VAE.

Figure 4 shows that although the VAE baseline is the best at preserving question meaning, it is also the worst at introducing variation to the output. HRQ-VAE better preserves the original question intent compared to the other systems while introducing more diversity than the VAE, as well as generating much more fluent output.

5.4 Ablations

To confirm that the hierarchical model allows for more expressive sketches, we performed two ablations. We compared to the full model using oracle sketches, so that code prediction performance was not a factor. We set the depth $D = 1$ and $K = 48$, giving equivalent total capacity to the full model ($D = 3, K = 16$) but without hierarchy. We also removed the initialisation scaling at lower depths, instead initialising all codebooks with the same scale. Table 6 shows that a non-hierarchical model with the same capacity is much less expressive.

We also performed two ablations against the model using predicted sketches; we removed depth dropout, so that the model is always trained on a full encoding. We confirm that learning the code-

books jointly with the encoder/decoder leads to a stronger model, by first training a model with a continuous Gaussian bottleneck (instead of the HRQ-VAE); then, we recursively apply k-means clustering (Lloyd, 1982), with the clustering at each level taking place over the residual error from all levels so far, analogous to HRQ-VAE. The results of these ablations shown in Table 6 indicate that our approach leads to improvements over all datasets.

6 Related Work

Hierarchical VAEs VQ-VAEs were initially proposed in computer vision (van den Oord et al., 2017), and were later extended to be 'hierarchical' (Razavi et al., 2019). However, in vision the term refers to a 'stacked' version architecture, where the output of one variational layer is passed through a CNN and then another variational layer that can be continuous (Vahdat and Kautz, 2020) or quantized (Williams et al., 2020; Liévin et al., 2019; Willetts et al., 2021). Unlike these approaches, we induce a *single* latent space that has hierarchical properties.

Other work has looked at using the properties of hyperbolic geometry to encourage autoencoders to learn hierarchical representations. Mathieu et al. (2019) showed that a model endowed with a Poincaré ball geometry was able to recover hierarchical structure in datasets, and Surís et al. (2021) used this property to deal with uncertainty in predicting events in video clips. However, their work was limited to continuous encoding spaces, and the hierarchy discovered was known to exist a priori.

Syntax-controlled Paraphrase Generation Prior work on paraphrasing has used retrieval techniques (Barzilay and McKeown, 2001), Residual LSTMs (Prakash et al., 2016), VAEs (Bowman et al., 2016), VQ-VAEs (Roy and Grangier, 2019) and pivot languages (Mallinson et al., 2017). Syntax-controlled paraphrase generation has seen significant recent interest, as a

means to explicitly generate diverse surface forms with the same meaning. However, most previous work has required knowledge of the correct or valid surface forms to be generated (Iyyer et al., 2018; Chen et al., 2019a; Kumar et al., 2020; Meng et al., 2021). It is generally assumed that the input can be rewritten without addressing the problem of predicting which template should be used, which is necessary if the method is to be useful. Hosking and Lapata (2021) proposed learning a simplified representation of the surface form using VQ, that could then be predicted at test time. However, the discrete codes learned by their approach are not independent and do not admit a known factorization, leading to a mismatch between training and inference.

7   Conclusion

We present a generative model of paraphrasing, that uses a hierarchy of discrete latent variables as a rough syntactic sketch. We introduce HRQ-VAE, a method for mapping these hierarchical sketches to a continuous encoding space, and demonstrate that it can indeed learn a hierarchy, with lower levels representing more fine-grained information. We apply HRQ-VAE to the task of paraphrase generation, representing the syntactic form of sentences as paths through a learned hierarchy, that can be predicted during testing. Extensive experiments across multiple datasets and a human evaluation show that our method leads to high quality paraphrases. The generative model we introduce has potential application for any natural language generation task; $\mathbf{z}_{sem}$ could be sourced from a sentence in a different language, from a different modality (e.g., images or tabular data) or from a task-specific model (e.g., summarization or machine translation). Furthermore, HRQ-VAE makes no assumptions about the type of space being represented, and could in principle be applied to a semantic space, learning a hierarchy over words or concepts.

Acknowledgements

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Yu Bao, Hao Zhou, Shujian Huang, Lei Li, Lili Mou, Olga Vechtomova, Xin-yu 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, Florence, Italy. Association for Computational Linguistics.

Regina Barzilay and Kathleen R. McKeown. 2001. Extracting paraphrases from a parallel corpus. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 50–57, Toulouse, France. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew M. Dai, Rafal Józefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *CoRR*, abs/1511.06349.

Yue Cao and Xiaojun Wan. 2020. DivGAN: Towards diverse paraphrase generation via diversified generative adversarial network. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2411–2421, Online. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019a. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019b. 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.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886, Copenhagen, Denmark. Association for Computational Linguistics.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1608–1618, Sofia, Bulgaria. Association for Computational Linguistics.

Yao Fu, Yansong Feng, and John P Cunningham. 2019. Paraphrase generation with latent bag of words. In *Advances in Neural Information Processing Systems*, volume 32, pages 13645–13656. Curran Associates, Inc.

Tanya Goyal and Greg Durrett. 2020. Neural syntactic preordering for controlled paraphrase generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 238–252, Online. Association for Computational Linguistics.

Tom Hosking and Mirella Lapata. 2021. Factorising meaning and form for intent-preserving paraphrasing. 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 1405–1418, Online. Association for Computational Linguistics.

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.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Karen Spärck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation*, 28:11–21.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona, Spain. Association for Computational Linguistics.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Transactions of the Association for Computational Linguistics*, 8(0):330–345.

Willem J. M. Levelt. 1993. *Speaking: From Intention to Articulation*. The MIT Press.

Valentin Liévin, Andrea Dittadi, Lars Maaløe, and Ole Winther. 2019. Towards hierarchical discrete variational autoencoders.

Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *ECCV*.

Zhe Lin and Xiaojun Wan. 2021. Pushing paraphrase away from original sentence: A multi-round paraphrase generation approach. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 1548–1557, Online. Association for Computational Linguistics.

S. Lloyd. 1982. Least squares quantization in pcm. *IEEE Transactions on Information Theory*, 28(2):129–137.

C Maddison, A Mnih, and Y Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *Proceedings of the international conference on learning Representations*. International Conference on Learning Representations.

Nitin Madnani and Bonnie J. Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 881–893, Valencia, Spain. Association for Computational Linguistics.

Randi C Martin, Jason E Crowther, Meredith Knight, Franklin P Tamborello II, and Chin-Lung Yang. 2010. Planning in sentence production: Evidence for the phrase as a default planning scope. *Cognition*, 116(2):177–192.

Emile Mathieu, Charline Le Lan, Chris J. Maddison, Ryota Tomioka, and Yee Whye Teh. 2019. Continuous hierarchical representations with poincaré variational auto-encoders. In *Advances in Neural Information Processing Systems*.

Yuxian Meng, Xiang Ao, Qing He, Xiaofei Sun, Qinghong Han, Fei Wu, Chun fan, and Jiwei Li. 2021. Conrpg: Paraphrase generation using contexts as regularizer.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Brussels, Belgium. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934, Osaka, Japan. The COLING 2016 Organizing Committee.

Ali Razavi, Aäron van den Oord, and Oriol Vinyals. 2019. *Generating Diverse High-Fidelity Images with VQ-VAE-2*. Curran Associates Inc., Red Hook, NY, USA.

Aurko Roy and David Grangier. 2019. Unsupervised paraphrasing without translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6033–6039, Florence, Italy. Association for Computational Linguistics.

Darsh Shah, Tao Lei, Alessandro Moschitti, Salvatore Romeo, and Preslav Nakov. 2018. Adversarial domain adaptation for duplicate question detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1056–1063, Brussels, Belgium. Association for Computational Linguistics.

Raphael Shu, Hideki Nakayama, and Kyunghyun Cho. 2019. Generating diverse translations with sentence codes. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1823–1827, Florence, Italy. Association for Computational Linguistics.

Casper Kaae Sønderby, Ben Poole, and Andriy Mnih. 2017. Continuous relaxation training of discrete latent variable image models. In *Beysian DeepLearning workshop, NIPS*, volume 201.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 38–42, Jeju Island, Korea. Association for Computational Linguistics.

Dídac Surís, Ruoshi Liu, and Carl Vondrick. 2021. Learning the predictability of the future.

Arash Vahdat and Jan Kautz. 2020. NVAE: A deep hierarchical variational autoencoder. In *Neural Information Processing Systems (NeurIPS)*.

Aaron van den Oord, Oriol Vinyals, and koray kavukcuoglu. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

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

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Matthew Willetts, Xenia Miscouridou, Stephen Roberts, and Chris Holmes. 2021. Relaxed-responsibility hierarchical discrete vaes.

Will Williams, Sam Ringer, Tom Ash, David MacLeod, Jamie Dougherty, and John Hughes. 2020. Hierarchical quantized autoencoders. In *Advances in Neural Information Processing Systems*, volume 33, pages 4524–4535. Curran Associates, Inc.

Ziang Xie, Sida I. Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y. Ng. 2017. Data noising as smoothing in neural network language models. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

A Hyperparameters

The hyperparameters given in Table 7 were selected by manual tuning, based on a combination of: (a) validation iBLEU scores with depth masking, (b) validation BLEU scores using oracle sketches, and (c) validation iBLEU scores using predicted syntactic codes.

The Gumbel temperature τ is decayed during training as a function of the step t, according to the following equation:

$$\tau(t) = \max(2 - \frac{2}{1 + e^{t/10000}}, 0.5). \quad (8)$$

Intuitively, this smoothly decays τ from an initial value of 2, with a half-life of 10k steps, to a minimum value of 0.5.

We use $\alpha = 0.8$ when calculating iBLEU, but as shown in Figure 5 our conclusions are not sensitive

Encoder/decoder	
Embedding dimension D	768
Encoder layers	5
Decoder layers	5
Feedforward dimension	2048
Transformer heads	8
Semantic/syntactic dim	192/594
Depth D	3
Codebook size K	16
Optimizer	Adam (Kingma and Ba, 2015)
Learning rate	0.01
Batch size	64
Token dropout	0.2 (Xie et al., 2017)
Decoder	Beam search
Beam width	4
Code predictor	
Num. hidden layers	2
Hidden layer size	3072

Table 7: Hyperparameter values used for our experiments.

to this value, and our model outperforms all comparison systems on all datasets for $0.7 \leq \alpha \leq 0.9$.

Models were trained on a single GPU, with training taking between one and three days depending on the dataset. We use SacreBLEU (Post, 2018) to calculate BLEU scores.

B Human Evaluation

Annotators were recruited from the UK and USA via Amazon Mechanical Turk, and were compensated for their time above a living wage in those countries. A full Participant Information Sheet was provided, and the study was approved by an internal ethics committee. Annotators were asked to rate the outputs according to the following criteria:

- Which system output is the most fluent and grammatical?

- To what extent is the meaning expressed in the original sentence preserved in the rewritten version, with no additional information added?

- Does the rewritten version use different words or phrasing to the original? You should choose the system that uses the most different words or word order.

C Exemplar Retrieval Process

Our approach requires exemplars during training to induce the separation between latent spaces. We follow the approach introduced by Hosking and

Input	How heavy is a moose?
Chunker output	How [heavy]$_{\text{ADVP}}$ is a [moose]$_{\text{NP}}$?
Template	How ADVP is a NP ?
Exemplar	How much is a surgeon's income?
Input	What country do parrots live in
Chunker output	What [country]$_{\text{NP}}$ do [parrots]$_{\text{NP}}$ [live]$_{\text{VP}}$ in ?
Template	What NP do NP VP in ?
Exemplar	What religion do Portuguese believe in?

Table 8: Examples of the exemplar retrieval process for training. The input is tagged by a chunker, ignoring stopwords. An exemplar with the same template is then retrieved from a different paraphrase cluster. Table reproduced with permission from Hosking and Lapata (2021).

Lapata (2021). During training, we retrieve exemplars $\mathbf{x}_{syn}$ from the training data following a process which first identifies the underlying syntax of $\mathbf{Y}$, and finds a question with the same syntactic structure but a different, arbitrary meaning. We use a shallow approximation of syntax, to ensure the availability of equivalent exemplars in the training data. An example of the exemplar retrieval process is shown in Table 8; we first apply a chunker (FlairNLP, Akbik et al., 2018) to $\mathbf{Y}$, then extract the chunk label for each tagged span, ignoring stopwords. This gives us the *template* that $\mathbf{Y}$ follows. We then select a question at random from the training data with the same template to give $\mathbf{x}_{syn}$. If no other questions in the dataset use this template, we create an exemplar by replacing each chunk with a random sample of the same type.

D Analysis of Code Properties

We define two features of sentences: (1) the presence of common auxiliary verbs that roughly indicate the tense of the sentence (present, future, etc.); and (2) the presence of different question or 'wh-' words[2]. We calculate the distributions of these features for each code q_d at different levels, with the results shown in Figure 6. Each column represents the distribution over the feature for a specific code. Figure 6a shows clear evidence that the sentences are (at least partly) clustered at the top level based on the verb used, while Figure 6b shows that level 2 encodes the question type.

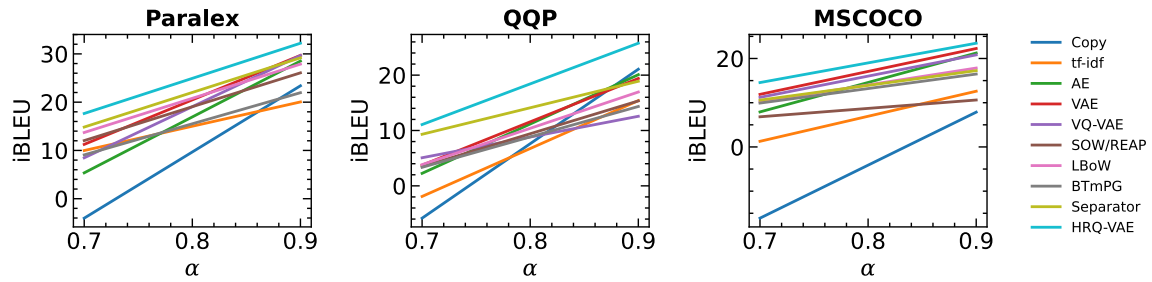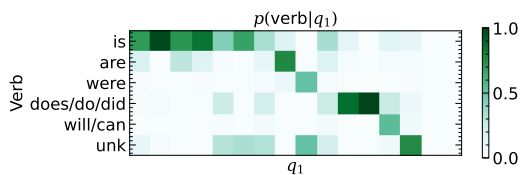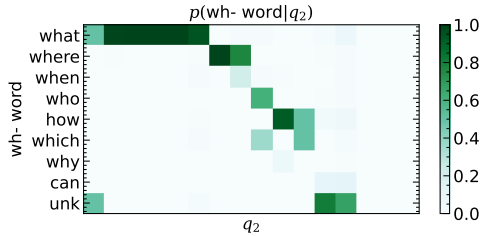[2]This analysis was performed for Paralex, which comprises entirely of questions.

Figure 5: iBLEU scores for all comparison systems, for a range of values of α.

(a) Distribution of verbs for each code within level 1.

(b) Distribution of wh- words for each code within level 2.

Figure 6: Plots showing the conditional distributions of two different sentence features, auxiliary verb and question type, for different values of the latent codes q_d. Each column represents the distribution over the feature for a specific code. The plots show that level 1 is a strong predictor of verb tense, and level 2 predicts question type, giving some insight into what syntactic features each level has learned to encode. We have re-ordered the columns of the plot to improve readability.