Factorising Meaning and Form for Intent-Preserving Paraphrasing

Tom Hosking **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 mlap@inf.ed.ac.uk

Abstract

We propose a method for generating paraphrases of English questions that retain the original intent but use a different surface form. Our model combines a careful choice of training objective with a principled information bottleneck, to induce a latent encoding space that disentangles meaning and form. We train an encoder-decoder model to reconstruct a question from a *paraphrase* with the same meaning and an *exemplar* with the same surface form, leading to separated encoding spaces. We use a Vector-Quantized Variational Autoencoder to represent the surface form as a set of discrete latent variables, allowing us to use a classifier to select a different surface form at test time. Crucially, our method does not require access to an external source of target exemplars. Extensive experiments and a human evaluation show that we are able to generate paraphrases with a better tradeoff between semantic preservation and syntactic novelty compared to previous methods.

1 Introduction

A paraphrase of an utterance is "an alternative surface form in the same language expressing the same semantic content as the original form" (Madnani and Dorr, 2010). For questions, a paraphrase should have the same intent, and should lead to the same answer as the original, as in the examples in Table 1. Question paraphrases are 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), as they allow a system to better identify the underlying intent of a user query.

Recent approaches to paraphrasing use information bottlenecks with VAEs (Bowman et al., 2016) or pivot languages (Wieting and Gimpel, 2018) to try to extract the semantics of an input utterance, before projecting back to a (hopefully different) surface form. However, these methods have little to no control over the preservation of the input meaning or variation in the output surface form. Other work has specified the surface form to be generated (Iyyer et al., 2018; Chen et al., 2019a; Kumar et al., 2020), but has so far assumed that the set of valid surface forms is known a priori.

How is a dialect different from a language?
The differences between language and dialect?
What is the difference between language and dialect?
What is the weight of an average moose?
Average weight of the moose?
How much do moose weigh?
How heavy is a moose?
What country do parrots live in?
In what country do parrots live?
Where do parrots naturally live?
What part of the world do parrots live in?

Table 1: Examples of question paraphrase clusters, drawn from Paralex (Fader et al., 2013). Each member of the cluster has essentially the same semantic *intent*, but a different *surface form*. Each cluster exhibits variation in word choice, syntactic structure and even question type. Our task is to generate these different surface forms, using only a single example as input.

In this paper, we propose SEPARATOR, a method for generating paraphrases that exhibit high variation in surface form while still retaining the original intent. Our key innovations are: (a) to train a model to reconstruct a target question from an input *paraphrase* with the same meaning, and an *exemplar* with the same surface form, and (b) to separately encode the form and meaning of questions as discrete and continuous latent variables respectively, enabling us to modify the output surface form while preserving the original question intent. Crucially, unlike prior work on syntax controlled paraphrasing, we show that we can generate diverse paraphrases of an input question at test time by inferring a different discrete syntactic encoding, without needing access to reference exemplars.

We limit our work to English questions for three reasons: (a) the concept of a paraphrase is more

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 1405–1418
August 1–6, 2021. ©2021 Association for Computational Linguistics

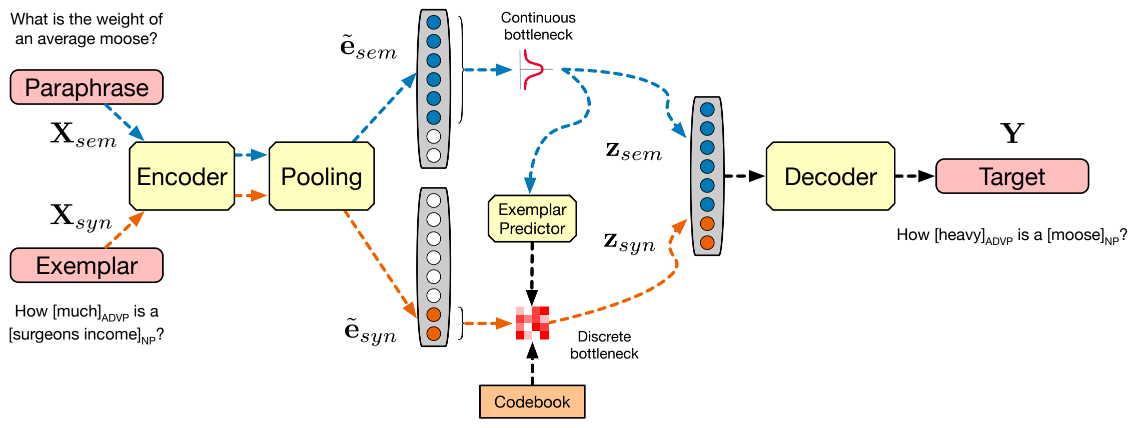


Figure 1: Overview of our approach. The model is trained to reconstruct a target question from one input with the same *meaning* and another input with the same *form*. This induces separate latent encoding spaces for meaning and form, allowing us to vary the output form while keeping the meaning constant. Using a discretized space for the syntactic encoding makes it tractable to predict valid surface forms at test time.

clearly defined for questions compared to generic utterances, as question paraphrases should lead to the same answer; (b) the space of possible surface forms is smaller for questions, making the task more achievable, and (c) better dataset availability. However, our approach does not otherwise make any assumptions specific to questions.

2 Problem Formulation

The task is to learn a mapping from an input question, represented as a sequence of tokens $\mathbf{X}$, to paraphrase(s) $\mathbf{Y}$ which have different *surface form* to $\mathbf{X}$, but convey the same *intent*.

Our proposed approach, which we call SEPARATOR, uses an encoder-decoder model to transform an input question into a latent encoding space, and then back to an output paraphrase. We hypothesize that a principled information bottleneck (Section 2.1) and a careful choice of training scheme (Section 2.2) lead to an encoding space that separately represents the intent and surface form. This separation enables us to paraphrase the input question, varying the surface form of the output by directly manipulating the syntactic encoding of the input and keeping the semantic encoding constant (Section 2.3). We assume access to reference *paraphrase clusters* during training (e.g., Table 1), sets of questions with different surface forms that have been collated as having the same meaning or *intent*.

Our model is a variant of the standard encoder-decoder framework (Cho et al., 2014), and consists of: (a) a vanilla Transformer sentence encoder (Vaswani et al., 2017), that maps an input question $\mathbf{X}$ to a multi-head sequence of encodings, $\mathbf{e}_{h,t} = \text{ENCODER}(\mathbf{X})$; (b) a principled choice of information bottleneck, with a continuous variational path and a discrete vector-quantized path, that maps the encoding sequence to a pair of latent vectors, $\mathbf{z}_{sem}, \mathbf{z}_{syn} = \text{BOTTLENECK}(\mathbf{e}_{h,t})$, represented in more detail in Figure 1; (c) a vanilla Transformer decoder, that attends over the latent vectors to generate a sequence of output tokens, $\hat{\mathbf{Y}} = \text{DECODER}(\mathbf{z}_{sem}, \mathbf{z}_{syn})$. The separation between $\mathbf{z}_{sem}$ and $\mathbf{z}_{syn}$ is induced by our proposed training scheme, shown in Figure 1 and described in detail in Section 2.2.

2.1 Model Architecture

While the encoder and decoder used by the model are standard Transformer modules, our bottleneck is more complex and we now describe it in more detail.

Let the encoder output be $\{\mathbf{e}_{h,1}, \ldots, \mathbf{e}_{h,|\mathbf{X}|}\} = \text{ENCODER}(\mathbf{X})$, where $\mathbf{e}_{h,t} \in \mathbb{R}^{D/H_T}$, $h \in 1, ..., H_T$ with H_T the number of transformer heads, $|\mathbf{X}|$ the length of the input sequence and D the dimension of the transformer. We first pool this sequence of encodings to a single vector, using the multi-head pooling described in Liu and Lapata (2019). For each head h, we calculate a distribution over time indexes $\alpha_{h,t}$ using attention:

$$\alpha_{h,t} = \frac{\exp a_{h,t}}{\sum_{t' \in |\mathbf{X}|} \exp a_{h,t'}}, \qquad (1)$$

$$a_{h,t} = \mathbf{k}_h^T \mathbf{e}_{h,t}, \qquad (2)$$

with $\mathbf{k}_h \in \mathbb{R}^{D/H}$ a learned parameter.

We then take a weighted average of a linear projection of the encodings, to give pooled output $\tilde{\mathbf{e}}_h$,

$$\tilde{\mathbf{e}}_h = \sum_{t' \in |\mathbf{X}|} \alpha_{h,t'} V_h \mathbf{e}_{h,t'}, \tag{3}$$

with $V_h \in \mathbb{R}^{D/H \times D/H}$ a learned parameter.

Transformer heads are assigned either to a *semantic* group H_{sem}, that will be trained to encode the intent of the input, $\tilde{\mathbf{e}}_{sem} = [\ldots; \tilde{\mathbf{e}}_h; \ldots], h \in H_{sem}$, or to a *syntactic* group H_{syn}, that will be trained to represent the surface form $\tilde{\mathbf{e}}_{syn} = [\ldots; \tilde{\mathbf{e}}_h; \ldots], h \in H_{syn}$ (see Figure 1).

The space of possible question intents is extremely large and may be reasonably approximated by a continuous vector space. However, the possible surface forms are discrete and smaller in number. We therefore use a Vector-Quantized Variational Autoencoder (VQ-VAE, van den Oord et al., 2017) for the syntactic encoding $\mathbf{z}_{syn}$, and model the semantic encoding $\mathbf{z}_{sem}$ as a continuous Gaussian latent variable, as shown in the upper and lower parts of Figure 1, respectively.

Vector Quantization Let q_h be discrete latent variables corresponding to the syntactic quantizer heads, $h \in H_{syn}$.[1] Each variable can be one of K possible latent codes, $q_h \in [0, K]$. The heads use distinct codebooks, $\mathbf{C}_h \in \mathbb{R}^{K \times D/H}$, which map each discrete code to a continuous embedding $\mathbf{C}_h(q_h) \in \mathbb{R}^{D/H}$. Given sentence $\mathbf{X}$ and its pooled encoding $\{\tilde{\mathbf{e}}_1, ..., \tilde{\mathbf{e}}_H\}$, we independently quantize the syntactic subset of the heads $h \in H_{syn}$ to their nearest codes from $\mathbf{C}_h$ and concatenate, giving the syntactic encoding

$$\mathbf{z}_{syn} = [\mathbf{C}_1(q_1); \ldots; \mathbf{C}_{|H_{syn}|}(q_{|H_{syn}|})]. \tag{4}$$

The quantizer module is trained through backpropagation using straight-through estimation (Bengio et al., 2013), with an additional loss term to constrain the embedding space as described in van den Oord et al. (2017),

$$\mathcal{L}_{cstr} = \lambda \sum_{h \in H_{syn}} \left\| \left(\tilde{\mathbf{e}}_h - \text{sg}(\mathbf{C}_h(q_h)) \right) \right\|_2, \tag{5}$$

where the stopgradient operator $\text{sg}(\cdot)$ is defined as identity during forward computation and zero on backpropagation, and λ is a weight that controls the strength of the constraint. We follow the soft EM and exponentially moving averages training approaches described in earlier work (Roy et al., 2018; Angelidis et al., 2021), which we find improve training stability.

[1]The number and dimensionality of the quantizer heads need not be the same as the number of transformer heads.

Variational Bottleneck For the semantic path, we introduce a learned Gaussian posterior, that represents the encodings as smooth distributions in space instead of point estimates (Kingma and Welling, 2014). Formally, $\phi(\mathbf{z}_h|\mathbf{e}_h) \sim \mathcal{N}(\boldsymbol{\mu}(\mathbf{e}_h), \boldsymbol{\sigma}(\mathbf{e}_h))$, where $\boldsymbol{\mu}(\cdot)$ and $\boldsymbol{\sigma}(\cdot)$ are learned linear transformations. To avoid vanishingly small variance and to encourage a smooth distribution, a prior is introduced, $\mathbf{p}(\mathbf{z}_h) \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. The VAE objective is the standard evidence lower bound (ELBO), given by

$$\begin{aligned} \text{ELBO} = -\text{KL}[\phi(\mathbf{z}_h|\mathbf{e}_h)||p(\mathbf{z}_h)] \\ + \mathbb{E}_\phi[\log p(\mathbf{e}_h|\mathbf{z}_h)]. \end{aligned} \tag{6}$$

We use the usual Gaussian reparameterisation trick, and approximate the expectation in Equation (6) by sampling from the training set and updating via backpropagation (Kingma and Welling, 2014). The VAE component therefore only adds an additional KL term to the overall loss,

$$\mathcal{L}_{KL} = -\text{KL}[\phi(\mathbf{z}_h|\mathbf{e}_h)||p(\mathbf{z}_h)]. \tag{7}$$

In sum, $\text{BOTTLENECK}(\mathbf{e}_{h,t})$ maps a sequence of token encodings to a pair of vectors $\mathbf{z}_{sem}, \mathbf{z}_{syn}$, with $\mathbf{z}_{sem}$ a continuous latent Gaussian, and $\mathbf{z}_{syn}$ a combination of discrete code embeddings.

2.2 Factorised Reconstruction Objective

We now describe the training scheme that causes the model to learn separate encodings for meaning and form: $\mathbf{z}_{sem}$ should encode only the intent of the input, while $\mathbf{z}_{syn}$ should capture any information about the surface form of the input. Although we refer to $\mathbf{z}_{syn}$ as the *syntactic encoding*, it will not necessarily correspond to any specific syntactic formalism. We also acknowledge that meaning and form are not completely independent of each other; arbitrarily changing the form of an utterance is likely to change its meaning. However, it is possible for the same intent to have multiple phrasings , and it is this 'local independence' that we intend to capture.

We create triples $\{\mathbf{X}_{sem}, \mathbf{X}_{syn}, \mathbf{Y}\}$, where $\mathbf{X}_{sem}$ has the same meaning but different form to $\mathbf{Y}$ (i.e., it is a paraphrase, as in Table 1) and $\mathbf{X}_{syn}$ is a question with the same form but different meaning

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 2: 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.

(i.e., it shares the same syntactic *template* as $\mathbf{Y}$), which we refer to as an *exemplar*. We describe the method for retrieving these exemplars in Section 2.3. The model is then trained to generate a target paraphrase $\mathbf{Y}$ from the semantic encoding $\mathbf{z}_{sem}$ of the input paraphrase $\mathbf{X}_{sem}$, and from the syntactic encoding $\mathbf{z}_{syn}$ of the exemplar $\mathbf{X}_{syn}$, as demonstrated in Figure 1.

Recalling the additional losses from the variational and quantized bottlenecks, the final combined training objective is given by

$$\mathcal{L} = \mathcal{L}_{\mathbf{Y}} + \mathcal{L}_{cstr} + \mathcal{L}_{KL}, \tag{8}$$

where $\mathcal{L}_{\mathbf{Y}}(\mathbf{X}_{sem}, \mathbf{X}_{syn})$ is the cross-entropy loss of teacher-forcing the decoder to generate $\mathbf{Y}$ from $\mathbf{z}_{sem}(\mathbf{X}_{sem})$ and $\mathbf{z}_{syn}(\mathbf{X}_{syn})$.

2.3 Exemplars

It is important to note that not all surface forms are valid or *licensed* for all question intents. As shown in Figure 1, our approach requires exemplars during training to induce the separation between latent spaces. We also need to specify the desired surface form at *test time*, either by supplying an exemplar as input or by directly predicting the latent codes. The output should have a different surface form to the input but remain fluent.

Exemplar Construction 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 2; 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.

We experimented with a range of approaches to determining question templates, including using part-of-speech tags and (truncated) constituency parses. We found that using chunks and preserving stopwords gave a reasonable level of granularity while still combining questions with a similar form. The templates (and corresponding exemplars) need to be granular enough that the model is forced to use them, but abstract enough that the task is not impossible to learn.

Prediction at Test Time In general, we do not assume access to reference exemplars at test time and yet the decoder must generate a paraphrase from semantic *and* syntactic encodings. Since our latent codes are separated, we can *directly* predict the syntactic encoding, without needing to retrieve or generate an exemplar. Furthermore, by using a discrete representation for the syntactic space, we reduce this prediction problem to a simple classification task. Formally, for an input question $\mathbf{X}$, we learn a distribution over licensed discrete codes $q_h, h \in \tilde{H}_{syn}$. We assume that the heads are independent, so that $p(q_1, \ldots, q_{\tilde{H}_{syn}}) = \prod_i p(q_i)$. We use a small fully connected network with the semantic and syntactic encodings of $\mathbf{X}$ as inputs, giving $p(q_h|\mathbf{X}) = \text{MLP}(\mathbf{z}_{sem}(\mathbf{X}), \mathbf{z}_{syn}(\mathbf{X}))$.

The network is trained to maximize the likelihood of all other syntactic codes licensed by each input. We calculate the discrete syntactic codes for each question in a paraphrase cluster, and minimize the cross-entropy loss of the network with respect to these codes. At test time, we set $q_h = \text{argmax}_{q'_h}[p(q'_h|\mathbf{X}_{test})]$.

3 Experimental Setup

Datasets We evaluate our approach on two datasets: Paralex (Fader et al., 2013), a dataset of question paraphrase clusters scraped from WikiAnswers; and Quora Question Pairs (QQP)[2] sourced from the community question answering forum Quora. We observed that a significant fraction of the questions in Paralex included typos or were ungrammatical. We therefore filter out any questions marked as non-English by a language detection

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

script (Lui and Baldwin, 2012), then pass the questions through a simple spellchecker. While this destructively edited some named entities in the questions, it did so in a consistent way across the whole dataset. There is no canonical split for Paralex, so we group the questions into clusters of paraphrases, and split these clusters into train/dev/test partitions with weighting 80/10/10. Similarly, QQP does not have a public test set. We therefore partitioned the clusters in the validation set randomly in two, to give us our dev/test splits. Summary statistics of the resulting datasets are given in Appendix B. All scores reported are on our test split.

Model Configuration Following previous work (Kaiser et al., 2018; Angelidis et al., 2021), our quantizer uses multiple heads ($H = 4$) with distinct codebooks to represent the syntactic encoding as 4 discrete categorical variables q_h, with $\mathbf{z}_{syn}$ given by the concatenation of their codebook embeddings $\mathbf{C}_h(q_h)$. We use a relatively small codebook size of $K = 256$, relying on the combinatoric power of the multiple heads to maintain the expressivity of the model. We argue that, assuming each head learns to capture a particular property of a template (see Section 4.3), the number of variations in *each property* is small, and it is only through combination that the space of possible templates becomes large.

We include a detailed list of hyperparameters in Appendix A. Our code is available at `http://github.com/tomhosking/separator`.

Comparison Systems We compare SEPARATOR against several related systems. These include a model which reconstructs $\mathbf{Y}$ only from $\mathbf{X}_{sem}$, with no signal for the desired form of the output. In other words, we derive both $\mathbf{z}_{sem}$ and $\mathbf{z}_{syn}$ from $\mathbf{X}_{sem}$, and *no separation* between meaning and form is learned. This model uses a continuous Gaussian latent variable for both $\mathbf{z}_{syn}$ and $\mathbf{z}_{sem}$, but is otherwise equivalent in architecture to SEPARATOR. We refer to this as the *VAE* baseline. We also experiment with a vanilla autoencoder or *AE* baseline by removing the variational component, such that $\mathbf{z}_{sem}, \mathbf{z}_{syn} = \tilde{\mathbf{e}}_{sem}, \tilde{\mathbf{e}}_{syn}$.

We include our own implementation of the VQ-VAE model described in Roy and Grangier (2019). They use a quantized bottleneck for *both* $\mathbf{z}_{sem}$ and $\mathbf{z}_{syn}$, with a large codebook $K = 64,000$, $H = 8$ heads and a residual connection within the quantizer. For QQP, containing only 55,611 training clusters, the configuration in Roy and Grangier (2019) leaves the model overparameterized and training did not converge; we instead report results for $K = 1,000$.

	Cluster type	
Encoding	**Paraphrase**	**Template**
$\mathbf{z}_{sem}$	0.943	0.096
$\mathbf{z}_{syn}$	0.952	0.092
$\mathbf{z}$	**0.960**	**0.096**

(a) VAE Baseline

	Cluster type	
Encoding	**Paraphrase**	**Template**
$\mathbf{z}_{sem}$	**0.944**	0.053
$\mathbf{z}_{syn}$	0.065	**0.866**
$\mathbf{z}$	0.307	0.849

(b) SEPARATOR

Table 3: Retrieval accuracies for each encoding for each cluster type. The VAE baseline is trained only on paraphrase pairs and receives no signal for the desired form of the output. SEPARATOR is able to learn separate encodings for meaning and form, with negligible loss in semantic encoding performance.

ParaNMT (Wieting and Gimpel, 2018) translates input sentences into a pivot language (Czech), then back into English. Although this system was trained on high volumes of data (including Common Crawl), the training data contains relatively few questions, and we would not expect it to perform well in the domain under consideration. 'Diverse Paraphraser using Submodularity' (DiPS; Kumar et al. 2019) uses submodular optimisation to increase the diversity of samples from a standard encode-decoder model. 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. We additionally implement a simple tf-idf baseline (Jones, 1972), retrieving the question from the training set with the highest similarity to the input. Finally, we include a basic copy baseline as a lower bound, that simply uses the input question as the output.

4 Results

Our experiments were designed to answer three questions: (a) Does SEPARATOR effectively factorize meaning and form? (b) Does SEPARATOR

Model	Paralex BLEU ↑	Paralex Self-BLEU ↓	Paralex **iBLEU** ↑	QQP BLEU ↑	QQP Self-BLEU ↓	QQP **iBLEU** ↑
Copy	37.10	100.00	−4.03	32.61	100.00	−7.17
VAE	40.26	66.12	8.35	19.36	35.29	2.96
AE	40.10	75.71	5.36	19.90	39.81	1.99
tf-idf	25.08	25.25	9.98	22.73	61.81	−2.63
VQ-VAE	40.26	65.71	8.47	16.19	26.15	3.43
ParaNMT	20.42	39.90	2.32	24.24	56.42	0.04
DiPS	24.90	29.58	8.56	18.47	32.45	3.19
SOW/REAP	33.09	37.07	12.04	12.64	24.19	1.59
LBoW	34.96	35.86	13.71	16.17	29.00	2.62
SEPARATOR	36.36	35.37	**14.84**	14.70	14.84	**5.84**
ORACLE	53.37	24.55	29.99	24.50	16.04	12.34

Table 4: Generation results, without access to oracle exemplars. Our approach achieves the highest iBLEU scores, indicating the best tradeoff between output diversity and fidelity to the reference paraphrases.

manage to generate diverse paraphrases (while preserving the intent of the input)? (c) What does the underlying quantized space encode (i.e., can we identify any meaningful syntactic properties)? We address each of these questions in the following sections.

4.1 Verification of Separation

Inspired by Chen et al. (2019b) we use a *semantic textual similarity* task and a *template detection* task to confirm that SEPARATOR does indeed lead to encodings $\{\mathbf{z}_{sem}, \mathbf{z}_{syn}\}$ in latent spaces that represent different types of information.

Using the test set, we construct clusters of questions that share the same meaning $\mathcal{C}_{sem}$, and clusters that share the same template $\mathcal{C}_{syn}$. For each cluster $C_q \in \{\mathcal{C}_{sem}, \mathcal{C}_{syn}\}$, we extract one question at random $\mathbf{X}_q \in C_q$, compute its encodings $\{\mathbf{z}_{sem}, \mathbf{z}_{syn}, \mathbf{z}\}$[3], and its cosine similarity to the encodings of all other questions in the test set. We take the question with maximum similarity to the query $\mathbf{X}_r, r = \text{argmax}_{r'}(\mathbf{z}_q.\mathbf{z}_{r'})$, and compare the cluster that it belongs to, C_r, to the query cluster $I(C_q = C_r)$, giving a *retrieval* accuracy score for each encoding type and each clustering type. For the VAE, we set $\{\mathbf{z}_{sem}, \mathbf{z}_{syn}\}$ to be the same heads of $\mathbf{z}$ as the separated model.

Table 3 shows that our approach yields encodings that successfully factorise meaning and form, with negligible performance loss compared to the VAE baseline; paraphrase retrieval performance using $\mathbf{z}_{sem}$ for the separated model is comparable to using $\mathbf{z}$ for the VAE.

[3] $\mathbf{z}$ refers to the combined encoding, i.e., $[\mathbf{z}_{sem}; \mathbf{z}_{syn}]$.

4.2 Paraphrase Generation

Automatic Evaluation While we have shown that our approach leads to disentangled representations, we are ultimately interested in generating diverse paraphrases for *unseen data*. That is, given some input question, we want to generate an output question with the same meaning but different form.

We use iBLEU (Sun and Zhou, 2012) as our primary metric, a variant of BLEU (Papineni et al., 2002; Post, 2018) that is penalized by the similarity between the output and the *input*,

$$\begin{aligned} \text{iBLEU} = \alpha \text{BLEU}(output, references) \\ -(1-\alpha)\text{BLEU}(output, input), \end{aligned} \quad (9)$$

where $\alpha = 0.7$ is a constant that weights the tradeoff between fidelity to the references and variation from the input. We also report the usual BLEU$(output, references)$ as well as Self-BLEU$(output, input)$. The latter allows us to examine whether the models are making trivial changes to the input. The Paralex test set contains 5.6 references on average per cluster, while QQP contains only 1.3. This leads to lower BLEU scores for QQP in general, since the models are evaluated on whether they generated the specific paraphrase(s) present in the dataset.

Table 4 shows that the Copy, VAE and AE models display relatively high BLEU scores, but achieve this by 'parroting' the input; they are good at reconstructing the input, but introduce little variation in surface form, reflected in the high Self-BLEU scores. This highlights the importance of considering similarity to both the references *and* to the input. The tf-idf baseline performs surprisingly

Input	What is the most known singer?
VAE	What is the most known singer?
DiPS	What was the most known famous singer?
SOW/REAP	What is the most famous singer?
Latent BoW	What is the most famous singer?
SEPARATOR	Who is the most famous singer in America?
Input	What is the income for a soccer player?
VAE	What is the salary for a soccer player?
DiPS	What is the median income in soccer?
SOW/REAP	What is US cer?
Latent BoW	What is the salary of a soccer [UNK]?
SEPARATOR	How much is a soccer players' salary?
Input	What has been the economic impact from Brexit referendum so far?
VAE	What has been the economic impact of Brexit referendum so far?
DiPS	What will be a impact of Brexit referendum?
SOW/REAP	How do I spend my virginity?
Latent BoW	How did Brexit referendum impact the Brexit referendum?
SEPARATOR	How much will the Brexit referendum cost?
Input	What are the basics I should know before learning Hadoop?
VAE	What are the basics should I know before learning Hadoop?
DiPS	How do I know before I want to learn Hadoop?
SOW/REAP	How can I know before learning Hadoop?
Latent BoW	What are the basics of learning Hadoop?
SEPARATOR	How much should I know before learning Hadoop?

Table 5: Examples of output generated by various approaches for a given input, from Paralex and QQP. SEPARATOR is able to generate questions with a different syntactic form to the input.

well on Paralex; the large dataset size makes it more likely that a paraphrase cluster with a similar meaning to the query exists in the training set.

The other comparison systems (in the second block in Table 4) achieve lower Self-BLEU scores, indicating a higher degree of variation introduced, but this comes at the cost of much lower scores with respect to the references. SEPARATOR achieves the highest iBLEU scores, indicating the best balance between fidelity to the references and novelty compared to the input. We give some example output in Table 5; while the other systems mostly introduce lexical variation, SEPARATOR is able to produce output with markedly different syntactic structure to the input, and can even change the question type while successfully preserving the original intent.

The last row in Table 4 (ORACLE) reports results when our model is given a valid exemplar to use directly for generation, thus bypassing the code prediction problem. For each paraphrase cluster, we select one question at random to use as input, and select another to use as the target. We retrieve a question from the training set with the same template as the target to use as an *oracle exemplar*. This represents an upper bound on our model's performance. While SEPARATOR outperforms existing methods, our approach to predicting syntactic codes (using a shallow fully-connected network) is relatively simple. SEPARATOR using oracle exemplars achieves by far the highest scores in Table 4, demonstrating the potential expressivity of our approach when exemplars are guaranteed to be valid. A more powerful code prediction model could close the gap to this upper bound, as well as enabling the generation of multiple diverse paraphrases for a single input question. However, we leave this to future work.

Human Evaluation In addition to automatic evaluation we elicited judgements from crowdworkers on Amazon Mechanical Turk. Specifically, they were shown a question and two paraphrases thereof (corresponding to different systems) and asked to select which one was preferred along three dimensions: the *dissimilarity* of the paraphrase compared to the original question, how well the paraphrase reflected the *meaning* of the original, and the *fluency* of the paraphrase (see Appendix C). We evaluated a total of 200 questions sampled equally from both Paralex and QQP, 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 according to Table 4 for our evaluation: SEPARATOR, DiPS (Kumar et al., 2019), Latent BoW (Fu et al., 2019) and VAE.

Figure 2 shows that although the VAE baseline is the best at preserving question meaning, it is also the worst at introducing variation to the output. SEPARATOR introduces more variation than the other systems evaluated and better preserves the original question intent, as well as generating significantly more fluent output (using a one-way ANOVA with post-hoc Tukey HSD test, $p<0.05$).

4.3 Analysis

When predicting latent codes at test time, we assume that the code for each head may be predicted independently of the others, as working with the full joint distribution would be intractable. We now examine this assumption as well as whether different encodings represent distinct syntactic proper-

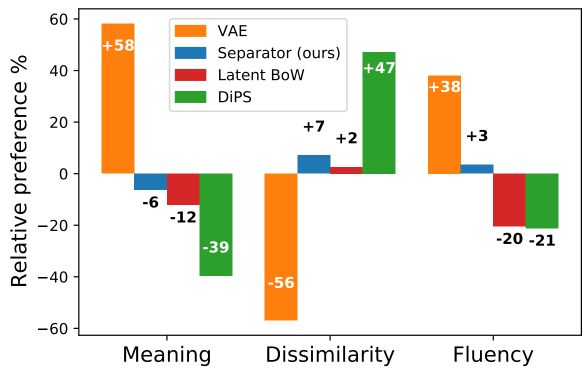


Figure 2: Results of our human evaluation. Although the VAE baseline is the best at preserving question meaning, it is the worst at introducing variation to the output. SEPARATOR offers the best balance between dissimilarity and meaning preservation, and is more fluent than both DiPS and Latent BoW.

ties. Following Angelidis et al. (2021), we compute the probability of a question property $f_1, f_2, \ldots$ taking a particular value a, conditioned by head h and quantized code k_h as

$$P(f_i|h,k_h) = \frac{\sum_{x\in\mathcal{X}} I(q_h(x)=k_h)I(f_i(x)=a)}{\sum_{x\in\mathcal{X}} I(q_h(x)=k_h)}, \quad (10)$$

where $I(\cdot)$ is the indicator function, and examples of values a are shown in Figure 3. We then calculate the mean entropy of these distributions, to determine how property-specific each head is:

$$\mathcal{H}_h = \frac{1}{K}\sum_{k_h}\sum_{a} P(a|h,k_h)\log P(a|h,k_h). \quad (11)$$

Heads with lower entropies are more predictive of a property, indicating specialisation and therefore independence. Figure 3 shows our analysis for four syntactic properties: head #2 has learned to control the high level output structure, including the question type or *wh- word*, and whether the question word appears at the beginning or end of the question. Head #3 controls which type of prepositional phrase is used. The length of the output is not determined by any one head, implying that it results from other properties of the surface form. Future work could leverage this disentanglement to improve the exemplar prediction model, and could lead to more fine-grained control over the generated output form.

In summary, we find that SEPARATOR successfully learns separate encodings for meaning and form. SEPARATOR is able to generate question

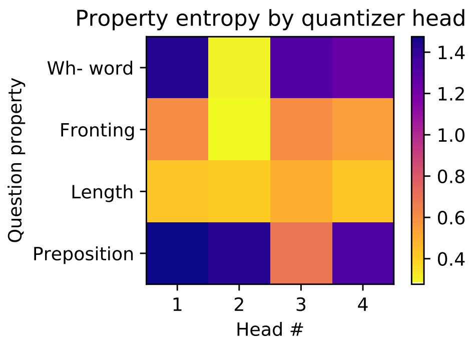


Figure 3: Predictive entropy by head for various question properties - lower entropy indicates higher predictive power.

paraphrases with a better balance of diversity and intent preservation compared to prior work. Although we are able to identify some high-level properties encoded by each of the syntactic latent variables, further work is needed to learn interpretable syntactic encodings.

5 Related Work

Paraphrasing Prior work on generating paraphrases has looked at extracting sentences with similar meaning from large corpora (Barzilay and McKeown, 2001; Bannard and Callison-Burch, 2005; Ganitkevitch et al., 2013), or identifying paraphrases from sources that are weakly aligned (Dolan et al., 2004; Coster and Kauchak, 2011).

More recently, neural approaches to paraphrasing have shown promise. Several models have used an information bottleneck to try to encode the semantics of the input, including VAEs (Bowman et al., 2016), VQ-VAEs (van den Oord et al., 2017; Roy and Grangier, 2019), and a latent bag-of-words model (Fu et al., 2019). Other work has relied on the strength of neural machine translation models, translating an input into a *pivot* language and then back into English (Mallinson et al., 2017; Wieting and Gimpel, 2018; Hu et al., 2019).

Kumar et al. (2019) use submodular function maximisation to improve the diversity of paraphrases generated by an encoder-decoder model. Dong et al. (2017) use an automatic paraphrasing system to rewrite inputs to a question answering system at inference time, reducing the sensitivity of the system to the specific phrasing of a query.

Syntactic Templates The idea of generating paraphrases by controlling the structure of the output has seen recent interest, but most work so far has assumed access to a template oracle. Iyyer et al.

(2018) use linearized parse trees as a template, then sample paraphrases by using multiple templates and reranking the output. Chen et al. (2019a) use a multi task objective to train a model to generate output that follows an input template. Their approach is limited by their use of automatically generated paraphrases for training, and their reliance on the availability of oracle templates. Bao et al. (2019) use a discriminator to separate spaces, but rely on noising the latent space to induce variation in the output form. Their results show good fidelity to the references, but low variation compared to the input. Goyal and Durrett (2020) use the artifically generated dataset ParaNMT-50m (Wieting and Gimpel, 2018) for their training and evaluation, which displays low output variation according to our results. Kumar et al. (2020) show strong performance using full parse trees as templates, but focus on generating output with the correct parse and do not consider the problem of template prediction.

Huang and Chang (2021) independently and concurrently propose training a model with a similar 'split training' approach to ours, but using constituency parses instead of exemplars, and a 'bag-of-words' instead of reference paraphrases. Their approach has the advantage of not requiring paraphrase clusters during training, but they do not attempt to solve the problem of template prediction and rely on the availability of oracle target templates.

Russin et al. (2020) modify the architecture of an encoder-decoder model, introducing an inductive bias to encode the structure of inputs separately from the lexical items to improve compositional generalisation on an artificial semantic parsing task. Chen et al. (2019b) use a multi-task setup to generate separated encodings, but do not experiment with generation tasks. Shu et al. (2019) learn discrete latent codes to introduce variation to the output of a machine translation system.

6 Conclusion

We present SEPARATOR, a method for generating paraphrases that balances high variation in surface form with strong intent preservation. Our approach consists of: (a) a training scheme that causes an encoder-decoder model to learn separated latent encodings, (b) a vector-quantized bottleneck that results in discrete variables for the syntactic encoding, and (c) a simple model to predict different yet valid surface forms for the output. Extensive experiments and a human evaluation show that our approach leads to separated encoding spaces with negligible loss of expressivity, and is able to generate paraphrases with a better balance of variation and semantic fidelity than prior methods.

In future, we would like to investigate the properties of the syntactic encoding space, and improve on the code prediction model. It would also be interesting to reduce the levels of supervision required to train the model, and induce the separation without an external syntactic model or reference paraphrases.

Acknowledgements

We thank our anonymous reviewers for their feedback. We are grateful to Stefanos Angelidis for many valuable discussions, and Hao Tang for their comments on the paper. This work was supported in part by the UKRI Centre for Doctoral Training in Natural Language Processing, funded by the UKRI (grant EP/S022481/1) and the University of Edinburgh. Lapata acknowledges the support of the European Research Council (award number 681760, "Translating Multiple Modalities into Text").

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.

Stefanos Angelidis, Reinald Amplayo, Yoshihiko Suhara, Xiaolan Wang, and Mirella Lapata. 2021. Extractive opinion summarization in quantized transformer spaces. *Transactions of the Association for Computational Linguistics*, 9(0):277–293.

Colin Bannard and Chris Callison-Burch. 2005. Paraphrasing with bilingual parallel corpora. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 597–604, Ann Arbor, Michigan. 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 Associ-*

ation for Computational Linguistics, pages 50–57, Toulouse, France. Association for Computational Linguistics.

Yoshua Bengio, N. Léonard, and Aaron C. Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *CoRR*, abs/1308.3432.

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.

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.

Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111, Doha, Qatar. Association for Computational Linguistics.

William Coster and David Kauchak. 2011. Simple English Wikipedia: A new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669, Portland, Oregon, USA. Association for Computational Linguistics.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 350–356, Geneva, Switzerland. COLING.

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.

Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2013. PPDB: The paraphrase database. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 758–764, Atlanta, Georgia. Association for Computational Linguistics.

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.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. *CoRR*, abs/1901.03644.

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.

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

Lukasz Kaiser, Aurko Roy, Ashish Vaswani, Niki Parmar, Samy Bengio, Jakob Uszkoreit, and Noam Shazeer. 2018. Fast decoding in sequence models using discrete latent variables. *CoRR*, abs/1803.03382.

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*.

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.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. 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 3609–3619, Minneapolis, Minnesota. Association for Computational Linguistics.

Yang Liu and Mirella Lapata. 2019. Hierarchical transformers for multi-document summarization. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5070–5081, Florence, Italy. Association for Computational Linguistics.

Marco Lui and Timothy Baldwin. 2012. langid.py: An off-the-shelf language identification tool. In *Proceedings of the ACL 2012 System Demonstrations*, pages 25–30, Jeju Island, Korea. Association for Computational Linguistics.

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

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.

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.

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

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, Belgium, Brussels. Association for Computational Linguistics.

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.

Aurko Roy, Ashish Vaswani, Arvind Neelakantan, and Niki Parmar. 2018. Theory and experiments on vector quantized autoencoders. *CoRR*, abs/1805.11063.

Jacob Russin, Jason Jo, Randall O'Reilly, and Yoshua Bengio. 2020. Compositional generalization by factorizing alignment and translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 313–327, Online. 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.

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.

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.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

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

Hyperparameters were selected by manual tuning, based on a combination of: (a) validation encoding separation, (b) validation BLEU scores using oracle exemplars, and (c) validation iBLEU scores using predicted syntactic codes.

Embedding dimension D	768
Encoder layers	5
Decoder layers	5
Feedforward dimension	2048
Transformer heads	8
Semantic/syntactic heads H_{sem}, H_{syn}	6/2
Quantizer heads $\tilde{H}_{syn}$	4
Quantizer codebook size K	256
Optimizer	Adam (Kingma and Ba, 2015)
Learning rate	0.005
Batch size	64
Token dropout	0.2 (Xie et al., 2017)
Decoder	Beam search
Beam width	4
Commitment weight λ	0.25
Code classifier	
Num. hidden layers	2
Hidden layer size	2712

Table 6: Hyperparameter values used for our experiments.

B Dataset Statistics

Summary statistics for our partitions of Paralex and QQP are shown in Table 7. Questions in QQP were 9.7 tokens long on average, compared to 8.2 for Paralex.

We also show the distribution of different question types in Figure 4; QQP contains a higher percentage of *why* questions, and we found that the questions tend to be more subjective compared to the predominantly factual questions in Paralex.

	Paralex		QQP	
	Clusters	Questions	Clusters	Questions
Train	222,223	1,450,759	55,611	138,965
Dev	27,778	183,273	5,255	12,554
Test	27,778	182,818	5,255	12,225

Table 7: Summary statistics for our cleaned version of (Fader et al., 2013), and our partitioning of QQP.

C Human Evaluation

Annotators were asked to rate the outputs according to the following criteria:

- Which system output is the most fluent and grammatical?

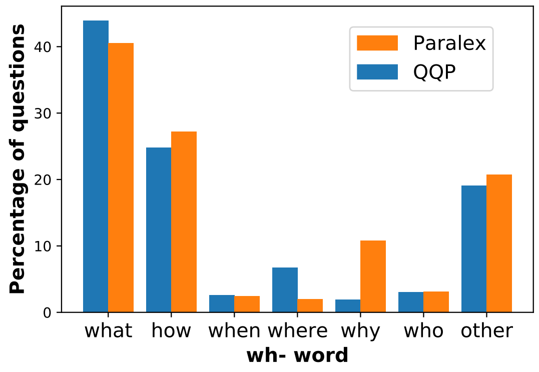


Figure 4: Distribution of wh- words for the datasets used in our experiments. QQP contains a much higher percentage of *why* questions.

- To what extent is the meaning expressed in the original question preserved in the rewritten version, with no additional information added? Which of the questions generated by a system is likely to have the same answer as the original?
- 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.

D Reproducibility Notes

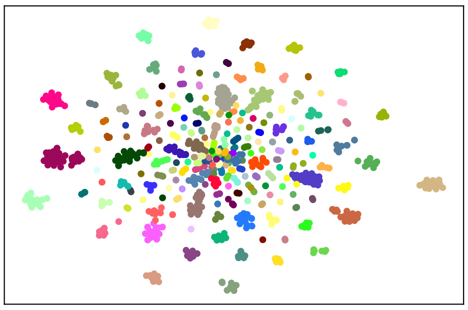
All experiments were run on a single Nvidia RTX 2080 Ti GPU. Training time for SEPARATOR was approximately 2 days on Paralex, and 1 day for QQP. SEPARATOR contains a total of 69,139,744 trainable parameters.

E Template Dropout

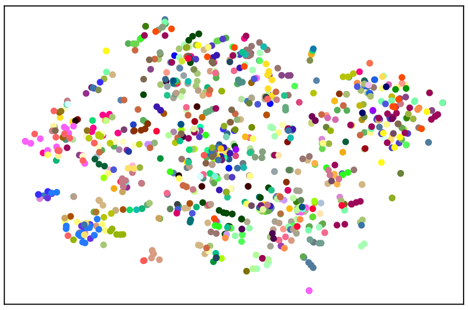
Early experiments showed that, while the model was able to separately encode meaning and form, the 'syntactic' encoding space showed little ordering. That is, local regions of the encoding space did not necessarily encode templates that *co-occurred* with each other in paraphrase clusters. We therefore propose *template dropout*, where exemplars $\mathbf{X}_{syn}$ are replaced with probability $p_{td} = 0.3$ by a question with a different template from the same paraphrase cluster. This is intended to provide the model with a signal about which templates are similar to each other, and thus reduce the distance between their encodings.

F Ordering of the Encoding Space

Figure 5 shows that the semantic encodings $\mathbf{z}_{sem}$ are tightly clustered by paraphrase, but the set of



(a) Semantic encodings



(b) Syntactic encodings

Figure 5: Visualisations of z_{sem} and z_{syn} using t-SNE (van der Maaten and Hinton, 2008), coloured by paraphrase cluster. The semantic encodings are clustered by meaning, as expected, but there is little to no local ordering in the syntactic space; valid surface forms of a particular question do not necessarily have syntactic encodings near to each other.

valid forms for each cluster overlaps significantly.

In other words, regions of licensed templates for each input are not contiguous, and naively perturbing a syntactic encoding for an input question is not guaranteed to lead to a valid template. Template dropout, described in Appendix E, seems to improve the arrangement of encoding space, but is not sufficient to allow us to 'navigate' encoding space directly. The ability to induce an ordered encoding space and introduce syntactic diversity by simply perturbing the encoding, would allow us to drop the template prediction network, and we hope that future work will build on this idea.

G Failure Cases

A downside of our approach is the use of an information bottleneck; the model must learn to compress a full question into a single, fixed-length vector. This can lead to loss of information or corruption, with the output occasionally repeating words or generating a number that is slightly different to the correct one, as shown in Table 8.

We also occasionally observe instances of the well documented posterior collapse phenomenon, where the decoder ignores the input encoding and generates a generic high probability sequence.

	Numerical error
Input	Replace starter on a 1988 Ford via?
Output	How do you replace a starter on a 1992 Ford?
	Repetition
Input	What brought about the organization of the Republican political party?
Output	What is the political party of the Republican party?
	Ignoring encoding
Input	What do Hondurans do for a living?
Output	What do Hondurans eat?

Table 8: Examples of failure modes.