Controllable Text Generation with Focused Variation

Lei Shu[1*] **, Alexandros Papangelis**[2] **, Yi-Chia Wang**[2] **, Gokhan Tur**[2]**,**

Hu Xu[1] **, Zhaleh Feizollahi**[2] **, Bing Liu**[1] **, Piero Molino**[3*]

[1]Department of Computer Science, University of Illinois at Chicago

[2]Uber AI,

[3]Stanford University and ML Collective

[1]shulindt@gmail.com, activebus@gmail.com, liub@uic.edu,

[2]al3x.papangelis@gmail.com, yichia.wang@gmail.com,

[2]gokhan.tur@ieee.org, zhaleh.feizollahi@gmail.com

[3]pmolino@cs.stanford.edu

Abstract

This work introduces Focused-Variation Network (FVN), a novel model to control language generation. The main problems in previous controlled language generation models range from the difficulty of generating text according to the given attributes, to the lack of diversity of the generated texts. FVN addresses these issues by learning disjoint discrete latent spaces for each attribute inside codebooks, which allows for both controllability and diversity, while at the same time generating fluent text. We evaluate FVN on two text generation datasets with annotated content and style, and show state-of-the-art performance as assessed by automatic and human evaluations.

1 Introduction

Recent developments in language modeling (Radford et al., 2019; Dai et al., 2019; Radford et al., 2018; Holtzman et al., 2020; Khandelwal et al., 2020) make it possible to generate fluent and mostly coherent text. Despite the quality of the samples, regular language models cannot be conditioned to generate language depending on attributes. Conditional language models have been developed to solve this problem, with methods that either train models given predetermined attributes (Shirish Keskar et al., 2019), use conditional generative models (Kikuchi et al., 2014; Ficler and Goldberg, 2017), fine-tune models using reinforcement learning (Ziegler et al., 2019), or modify the text on the fly during generation (Dathathri et al., 2020).

As many researchers noted, injecting style into natural language generation can increase the naturalness and human-likeness of text by including pragmatic markers, characteristic of oral language (Biber, 1991; Paiva and Evans, 2004; Mairesse and Walker, 2007). Text generation with style-variation has been explored as a special case of conditional language generation that aims to map attributes such as the informational content (usually structured data representing meaning like frames with keys and values) and the style (such as personality and politeness) into one of many natural language realisations that conveys them (Novikova et al., 2016, 2017; Wang et al., 2018). As the examples in Table 1 show, for one given content frame there can be multiple realisations.When a style (a personality trait in this case) is injected, the text is adapted to that style (words in red) while conveying the correct informational content (words in blue). A key challenge is to generate text that respects the specified attributes while at the same time generating diverse outputs, as most existing methods fail to correctly generate text according to given attributes or exhibit a lack of diversity among different samples, leading to dull and repetitive expressions.

Conditional VAEs (CVAE) (Sohn et al., 2015) and their variants have been adopted for the task and are able to generate diverse texts, but they suffer from posterior collapse and do not strictly follow the given attributes because their latent space is pushed towards being a Gaussian distribution irrespective of the different disjoint attributes, conflating the given content and style.

An ideal model would learn a separate latent space that focuses on each attribute independently. For this purpose, we introduce a novel natural language generator called Focused-Variation Network (FVN)[1]. FVN extends the Vector-Quantised VAE (VQ-VAE) (van den Oord et al., 2017), which is non-conditional, to allow conditioning on attributes (content and style). Specifically, FVN: (1) models two disjoint codebooks for content and style respectively that memorize input text vari-

[*]Work done while at Uber AI Labs.

[1]The code is available at https://leishu02.github.io/

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 3805–3817
November 16 - 20, 2020. ©2020 Association for Computational Linguistics

CMR	Name[Fitzbillies], EatType[pub], Food[Italian], CustomerRating[decent], Area[Riverside], FamilyFriendly[No], Near[The Sorrento], PriceRange[Moderate]
Text 1	Fitzbillies is a pub with a decent rating. It is a moderately priced Italian restaurant in riverside near The Sorrento. It is not family-friendly.
Delex. Text 1	Name_SLOT is a EatType_SLOT with a CustomerRating_SLOT rating. It is a PriceRange_SLOT priced Food_SLOT restaurant in Area_SLOT near Near_SLOT. It is FamilyFriendly_SLOT.
Agreeable	Let's see what we can find on *Name_SLOT.* I see, well, *it is an EatType_SLOT with a CustomerRating_SLOT rating, also it is a PriceRange_SLOT priced Food_SLOT restaurant in Area_SLOT and near Near_SLOT, also it is FamilyFriendly_SLOT,* you see?
Disagreeable	I mean, everybody knows that *Name_SLOT is an EatType_SLOT with a CustomerRating_SLOT rating. It is a PriceRange_SLOT priced Food_SLOT restaurant in Area_SLOT near Near_SLOT. It is FamilyFriendly_SLOT.*
Delex. Text 2	Name_SLOT is a Food_SLOT place near Near_SLOT in Area_SLOT and PriceRange_SLOT priced. It has a CustomerRating_SLOT rating. It is an EatType_SLOT and FamilyFriendly_SLOT kid friendly.
Conscientious	Let's see what we can find in *Name_SLOT.* Emm ... *it is a Food_SLOT place near Near_SLOT in Area_SLOT and PriceRange_SLOT priced. It has a CustomerRating_SLOT rating. It is an EatType_SLOT and FamilyFriendly_SLOT.*
Unconscientious	Oh god yeah, I don't know. *Name_SLOT is a Food_SLOT place near Near_SLOT in Area_SLOT and PriceRange_SLOT priced. It has a CustomerRating_SLOT rating. It is an EatType_SLOT and FamilyFriendly_SLOT kid friendly.*

Table 1: Text generation with focused variations (underlined red denotes personality, italics blue denotes content). The styles are personality traits (dis/agreeable, un/conscientious, extrovert). The content meaning representation and neutral text (Text 1 and 2) are shown at the top. When given a style, the generated text *strictly* follows it. Delex denotes delexicalised text.

ations; (2) further controls the conveyance of attributes by using content and style specific encoders and decoders; (3) computes disjoint latent space distributions that are conditional on the content and style respectively, which allows to sample latent representations in a focused way at prediction time. This choice ultimately helps both attribute conveyance and variability. As a result, FVN can preserve the diversity found in training examples as opposed to previous methods that tend to cancel out diverse examples. FVN's disjoint modeling of content and style increases the conveyance of the generated text, while at the same time generating more natural and fluent text.

We tested FVN on two datasets, PersonageNLG (Oraby et al., 2018) and E2E (Dušek et al., 2020) that consist of content-utterance pairs with personality labels in the first case, and the experimental results show that it outperforms previous state-of-the-art methods. A human evaluation further confirms that the naturalness and conveyance of FVN generated text is comparable to ground truth data.

2 Related Work

Our work is related to CVAE based text generation (Bowman et al., 2016; Shen et al., 2018; Zhang et al., 2019), where the goal is to control a given attribute of the output text (for example, style) by providing it as additional input to a regular VAE. For instance, the controlled text generation method proposed by Hu et al. (2017) extends VAE and focuses on controlling attributes of the generated text like sentiment and style. Differently from ours, this method does not focus on generating text from content meaning representation (CMR) or on diversity of the generated text. (Song et al., 2019) use a memory augmented CVAE to control for persona, but with no control over the content.

The works of (Oraby et al., 2018; Harrison et al., 2019; Oraby et al., 2019) on style-variation generators adopt sequence-to-sequence based models and use human-engineered features (Juraska and Walker, 2018) (e.g. personality parameters or syntax features) as extra inputs alongside the content and style to control the generation and enhance text variation. However, using human-engineered features is labor-intensive and, as it is not possible to consider all possible feature combinations, performance can be sub-optimal. In our work we instead rely on codebooks to memorize textual variations.

There is a variety of works that address the problem of incorporating knowledge or structured data into the generated text (for example, entities retrieved from a knowledge base) (Ye et al., 2020), or that try generate text that is in line with some given story (Rashkin et al., 2020). None of these works focuses specifically on generating text that conveys content while at the same time controlling style. Last, there are works such as (Rashkin et al., 2018) that focus on generating text consistent with an emotion (aiming to create an empathetic agent) without, however, directly controlling the content.

3 Methodology

Our proposed FVN architecture (Figure 1) has the goal to generate diverse texts that respect every attribute provided as controlling factor. We describe a specific instantiation where the attributes

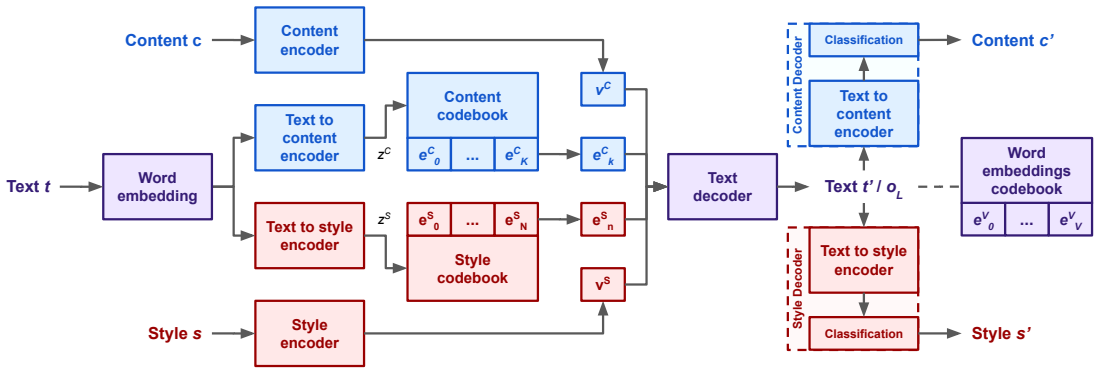

Figure 1: Focused-Variation Network (FVN) has four encoders (text-to-content encoder, text-to-style encoder, content encoder and style encoder), two codebooks (for content and style), and one text decoder. The training data contains ground-truth text with associated content and style. The text decoder uses v^C and v^S, latent vectors of content and style, as well as the latent vectors e^C_k and e^S_n from codebooks (the nearest to the z^C and z^S vectors produced by the text-to-content and text-to-style encoders) to generate text back. To further control the content and style of the generated text, we feed the o_L output vectors of the generated text t' to text encoders (content and style). o_L are aligned to a word embedding codebook.

are content (a frame CMR containing slots keys and values) and style (personality traits). However, the same architecture can be used with additional attributes and / or with different types of content attributes (structured data tables and knowledge graphs for instance) and style attributes (linguistic register, readability, and many others). To encourage conveyance of the generated texts, FVN learns disjoint discrete content- and style-focused representation codebooks inspired by VQ-VAE as extra information along with the representations of intended content and style, which avoids the posterior collapse problem of VAEs.

During training, FVN receives as input an intended content c and style s as well as a reference text t. The reference text is passed through two encoders (text-to-content and text-to-style), while content and style are encoded with a content encoder and a style encoder. The text-to-content encoder maps input text t into a content latent vector z^C and the text-to-style encoder maps the input text t into a latent style vector z^S. The closest vectors to z^C and z^S from the content codebook e^C and style codebook e^S, e^C_k and e^S_n, are selected. The content encoder encodes the intended content frame into a latent vector v^C and the style encoder encodes the intended style into a latent vector v^S . A text decoder then receives e^C_k, e^S_n, v^C and v^S and generates the output text t'. The generated text is subsequently fed to a content and a style decoder that predict the intended content and style.

At prediction time (Figure 2), only content c and style s are given, and in order to obtain e^C_k and e^S_n without an input text, we A) collect a distribution over the codebook indices by counting, for each training datapoint containing a specific value for c and s, the amount of times a specific index is used, and B) sample e^C_k and e^S_n from these frequency distributions. These disjoint distributions allow the model to focus on specific content and style by using them for conditioning and the sampling allows for variation, hence the name of focused variation. v^C and v^S obtained from the content and style encoders and the sampled e^C_k and e^S_n are provided to the text generator that generates t'.

The rest of this section will detail each component and the training and prediction processes.

3.1 Encoding and Codebooks

As shown in Figure 1, FVN uses four encoders and one decoder during training: the text-to-content encoder $\text{Enc}^{TC}(\cdot)$, the text-to-style encoder $\text{Enc}^{TS}(\cdot)$, the content encoder $\text{Enc}^{C}(\cdot)$, the style encoder $\text{Enc}^{S}(\cdot)$, and the text decoder $\text{Dec}(\cdot)$.

Text-to-* encoders The text-to-content encoder $\text{Enc}^{TC}(\cdot)$ encodes a text t to a dense representation $z^C \in \mathcal{R}^D$ while the text-to-style encoder $\text{Enc}^{TS}(\cdot)$ encodes a text t to a dense representation $z^S \in \mathcal{R}^D$: $z^C = \text{Enc}^{TC}(t)$ and $z^S = \text{Enc}^{TS}(t)$.

In order to learn disjoint latent spaces for the different attributes we want to model, we train two codebooks, one for content $e^C \in \mathcal{R}^{K \times D}$ and one for style $e^S \in \mathcal{R}^{N \times D}$. They are shown as $[e^C_1, \ldots, e^C_K]$ and $[e^S_1, ..., e^S_N]$ in Figure 1.

These two codebooks are used to memorize the latent vectors for text-to-content variation and text-to-style variation learned during training. Instead of using the z^C and z^S vectors as inputs to the de-

coder, we find their nearest latent vectors in the codebooks e_k^C and e_n^S and use those nearest latent vectors for decoding the text instead of the original encoded dense representation. Formally, $k = \text{argmin}_i \|z^C - e_i^C\|_2$ and $n = \text{argmin}_j \|z^S - e_j^S\|_2$.

Like in VQ-VAE, we use the $l2$-norm error to move the latent vectors in the codebooks e towards the same space of the encoder outputs z:

$$\mathcal{L}_{VQ}^C = \|sg(z^C) - e_k^C\|_2^2 + \beta^C \|z^C - sg(e_k^C)\|_2^2, \quad (1)$$

$$\mathcal{L}_{VQ}^S = \|sg(z^S) - e_n^S\|_2^2 + \beta^S \|z^S - sg(e_n^S)\|_2^2, \quad (2)$$

where $sg(\cdot)$ stands for the stop gradient operator.

Style and content encoders The content encoder encodes a CMR c treating it as a sequence of tokens and producing a matrix $V^C \in \mathcal{R}^{L' \times D}$, where L' is the length of c, from which the last element $v^C \in \mathcal{R}^D$ is returned. The style encoder encodes a style s and obtains a dense representation $v^S \in \mathcal{R}^D$ selecting the last element of the matrix $V^S \in \mathcal{R}^{L'' \times D}$. Ultimately, $v^C = \text{Enc}^M(m)$ and $v^S = \text{Enc}^S(s)$.

Both sets of vectors, e and v are needed as the former learn to memorize the encoded inputs z, while the latter learn regularities in the attributes.

3.2 Text Decoder

The decoder takes the e_k^C, e_n^S, v^C and v^S, which encode content and style, as input and decodes text t'. We use an LSTM network to model our decoder and provide the initial hidden state h_0 and initial cell state c_0. The initial hidden state is the concatenation of e_k^C and e_n^S, while the initial cell state is the concatenation of v^C and v^S: $c_0 = v^C \circ v^S$ and $h_0 = e_k^C \circ e_n^S$.

When we decode the l-th word, we encode the previous word t'_{l-1} and pay attention to the encoded sequence of content v^C and style v^S using the last hidden state as a query. Since both content and style are sequences of words, the attention mechanism can help figure out which part of them is important for decoding the current word. We concatenate the embedded previous output word and the attention output as the input for LSTM x_l. The LSTM updates the hidden state, cell state and produces an output vector $g_l \in \mathcal{R}^{2D}$. Since we want to feed the generated text back to text encoders for additional control, we reduce g_l to a word embedding dimension vector o_l by a linear transformation. Finally, we map o_l to the size of the vocabulary and apply softmax to obtain a probability distribution over the vocabulary.

$$x_l = \text{Emb}(t'_{l-1}) \circ \text{Attn}(h_{l-1}, V^C \circ V^S), \quad (3)$$

$$g_l, (h_l, c_l) = \text{LSTM}\Big(x_l, (h_{l-1}, c_{l-1})\Big), \quad (4)$$

$$o_l = W_{emb} \cdot g_l + b_{emb}, \quad (5)$$

$$P(t'_l) = \text{softmax}(W_V \cdot o_l + b_V). \quad (6)$$

The loss for text decoding is the sum of cross entropy loss of each word os $\mathcal{L}_{Dec} = -\sum_l \log P(t'_l)$.

3.3 Content and Style Decoders

To ensure the generated text t' conveys the correct content and style, we feed them to content and style decoders to perform backward prediction tasks that better control the generator. The decoders contain two components: we first reuse the text-to-content and text-to-style encoders to encode the embedded predicted text o_L and obtain latent representations z'^C and z'^S, and then we classify them to predict content c' and style s', as shown in the right side of Figure 1: $z'^C = \text{Enc}^{TC}(o_L)$ and $z'^S = \text{Enc}^{TS}(o_L)$. $\text{Enc}^{TC}(\cdot)$ and $\text{Enc}^{TS}(\cdot)$ denote the same text-to-content and text-to-style encoders we defined previously. This design is inspired by work on text style transfer (dos Santos et al., 2018).

Both z' vectors and e vectors are used by two classification heads F^C (multi-label) and F^S (multi-class) for predicting content and style respectively in order to force those vectors to encode attribute information. We use g to denote the g-th element in the set of possible key-value pairs in the CMR and $m(\cdot)$ to represent an indicator function that returns whether the g-th element is in the ground-truth CMR.

$$P\Big(y_z^C(g) = m(g)\Big) = F^C(z'^C), \quad (7)$$

$$P(y_z^S = s) = F^S(z'^S), \quad (8)$$

$$P\Big(y_e^C(g) = m(g)\Big) = F^C(e_k^C), \quad (9)$$

$$P(y_e^S = s) = F^S(e_n^S). \quad (10)$$

The loss for training the two prediction heads is:

$$\mathcal{L}_{CTRL} = -\sum_g \log P\Big(y_e^C(g) = m(g)\Big) - \log P(y_e^S = s) - \sum_g \log P\Big(y_z^C(g) = m(g)\Big) - \log P(y_z^S = s). \quad (11)$$

Finally, we also adopt vector quantization by mapping each generated word's representation o_l to the word embedding $e^V \in \mathcal{R}^{|V| \times D}$ to map the output of the decoder and the input of text encoders in the same space. This is needed because the text-to-* encoders expect as input text embedded using

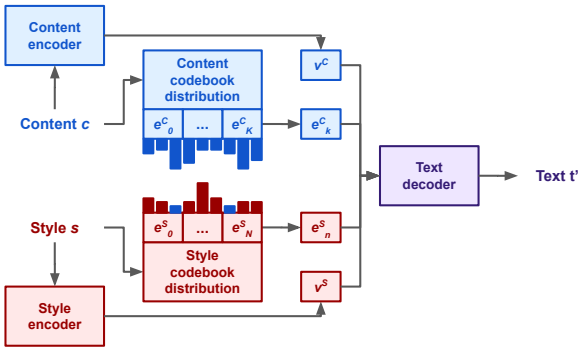

Figure 2: At prediction time we encode c and d with encoders to obtain v^C and v^S and we select e_k^C by sampling $k \sim P(K|C = c)$ and e_n^S by sampling $n \sim P(N|S = s)$. Those four vectors are provided as input to the text decoder to generate text.

word embeddings, but in this case we are providing o_L as input, and without this vector quantization loss, o_L will not be in the same space of the embeddings. As a result, there is another VQ loss: $\mathcal{L}_{VQ}^V = \|sg(o_l) - e_v^V\|_2^2 + \beta^V \|o_l - sg(e_v^V)\|_2^2$.

The total loss minimized during training is the sum of the losses for decoding the text, predicting the content and style, the VQ-loss from two codebooks, and the VQ-loss for word embedding: $\mathcal{L} = \mathcal{L}_{Dec} + \mathcal{L}_{CTRL} + \mathcal{L}_{VQ}^C + \mathcal{L}_{VQ}^S + \mathcal{L}_{VQ}^V$.

3.4 Prediction

The whole prediction process is depicted in Figure 2. The trained text decoder expect four inputs: v^C, v^S, e_k^C, and e_n^S. At prediction time, only content c and style s are given. We can obtain v^C, v^S by providing c and s to their respective encoder, but we also need to obtain e_k^C and e_n^S without input text. At the end of the training phase, we map each content $c \in C$ and style $s \in S$ to the indices in the e^C and e^S codebooks by first obtaining z^S and z^C vectors from the training data associated with c and s, we find the index of the closest codebooks vectors by $\text{argmin}_k \|e_k^C - z^C\|_2$ and $\text{argmin}_n \|e_n^S - z^S\|_2$ and count how many times each index $k \in K$ was the closes to each $c \in C$ and likewise for indices $n \in N$ for each $s \in S$. By normalizing the counts, we obtain a distribution $P(K|C)$ for content and a distribution $P(N|S)$ for style. The construction of the two distributions is performed only once at the end of the training phase.

To obtain e_k^C at prediction time, we select the k vector of the codebook by sampling $k \sim P(K|C = c)$ and likewise to obtain e_n^S with $n \sim P(N|S = s)$. Sampling from those distributions allows to both focus on specific content and style disjointly by conditioning on them, while at the same time allowing variability because of the sampling (we refer to this procedure as focused variation). v^C, v^S, e_k^C, and e_n^S are finally provided as inputs to the decoder to generate the text t'. Content and style decoders mentioned in the training section are not needed for prediction.

Module	Layers (in, out)
content codebook	Emb(K, D)
style/slot-value codebook	Emb(N, D)
text-to-content encoder	Emb($\|V\|$, D), Bi-LSTM(D, D)
text-to-style encoder	Emb($\|V\|$, D), Bi-LSTM(D, D)
content encoder	Emb($\|V\|$, D), Bi-LSTM(D, D)
style encoder	Emb($\|V\|$, D), Bi-LSTM(D, D)
content decoder	Dense(D, D/2), Dense(D/2, 8)
style decoder	Dense(D, D/2), Dense(D/2, 5)
slot-value decoder	Dense(D, D/2), Dense(D/2, 36)
text decoder	Emb($\|V\|$, D), LSTM(D,2D), Attn(2D, 2D) Dense(2D, D), Dense(D,$\|V\|$)

Table 2: Details of Modules in FVN: $D = 300$, $K = 512$, $N = |V|$

4 Experiments

To test the capability of FVN to generate diverse texts that convey the content while adopting a certain style, we use the PersonageNLG text generation dataset for dialogue systems that contains CMR and style annotations. To test if FVN can convey the content (both slots and values) correctly on an open vocabulary, with complex syntactic structures and diverse discourse phenomena, we use the End-2-End Challenge dataset (E2E), a text generation dataset for dialogue systems that is annotated with CMR.

4.1 Datasets and Baselines

PersonageNLG contains 88,855 training and 1,390 test examples. We reserve 10% of the train set for validation. There are 8 slots in the CMR and 5 kinds of style: agreeable, disagreeable conscientious, unconscientious, and extravert personality traits. The styles are evenly distributed in both train and test sets. All slots' values are delexicalized. We model the focused variation distribution of the content by jointly modeling the presence of slot names in the CMR, e.g. $P(K|PriceRange \in c \; and \; FoodType \in c)$, because there are no slot values. Style is modeled as a single categorical variable, e.g. $P(N|s = Agreable)$.

E2E contains 42,061 training examples (4,862 CMRs), 4,672 development examples (547 CMRs) and 4,693 test examples (630 CMRs). Like in

the PersonageNLG dataset, there are 8 slots in the CMR. Each CMR has up to 5 realisations (references) in natural language. Differently from PersonageNLG, the CMRs in the different splits are disjoint and the texts are lexicalized. Following the challenge guidelines (Dušek et al., 2018), we delexicalized only 'name' and 'near' keeping the remaining slots' values. Since the E2E dataset does not have style annotations but has lexicalized texts, we model the CMR in the same way we did for PersonageNLG, but we replace the style codebook with a slot-value codebook that help the text decoder generating the slot values in the CMR. We build the focused variation distribution for every slot-value independently over the codebook indices, e.g. $P(N|s = PriceRange[high])P(N|s = FoodType[French])$.... During prediction we sample codes for each slot value in the CMR and use their average to condition text decoding. This is particularly useful when the surface forms in the output text are not the slot values themselves, e.g. when "PriceRange[high]" should be generated as "expensive" rather than "high".

We use NLTK (Bird et al., 2009) to tokenize each sentence and de-lexicalize the text as described in (Dušek and Jurcicek, 2016a). We use 300-dimensional GloVe embeddings (Pennington et al., 2014) trained on 840B words. Words not in GloVe are initialized as the averaged embeddings of all other embeddings plus a small amount of random noise to make them different from each other. The details of each module in the FVN are listed in Table 2. We set $D = 300$, $K = 512$, $N = |V|$. The encoders are three-layer stacked Bi-LSTM and the text decoder is one-layer LSTM. The style/slot-value codebook is initialized as pre-trained word embedding. The content codebook is uniformed initialized in the range of $[-1/K, 1/K]$. We use the Adam optimizer (Kingma and Ba, 2015) with a learning rate of 0.001 for minimizing the total loss. More dataset details are shown in Appendix A Table 13 and Table 14.

We compare our proposed model against the best performing models in both datasets. All of them are sequence-to-sequence based models. For PersonageNLG, **TOKEN**, **CONTEXT** (from (Oraby et al., 2018)) are variants of the TGEN (Novikova et al., 2017) architecture, while **token-m*** and **context-m*** are from (Harrison et al., 2019) (which adopt OpenNMT-py (Klein et al., 2017) as the basic encoder-decoder architecture). token-* baselines use a special style token to provide style information while context-* baselines use 36 human defined pragmatic and aggregation-based features to provide style information '-m*' indicates variants of how the style information is injected into the encoder and the decoder. For the E2E challenge dataset, **TGEN** (Novikova et al., 2017), **SLUG** (Juraska et al., 2018), and **Thomson Reuters NLG** (Davoodi et al., 2018; Smiley et al., 2018) are the best performing models. They have different architectures, re-rankers, beam search and data augmentation strategies. More details are provided in Appendix B.

The results of the baselines (Oraby et al., 2018; Harrison et al., 2019) are taken from their original papers, but it's unclear if they were evaluated using a single or multiple references (for this reason they are marked with †), but since these models are not dependent on sampling from a latent space, we would not expect that to change performance.

We also compare to conditional VAEs: **CVAE** implements the conditional VAE (Sohn et al., 2015) framework. **Controlled CVAE** implements the controlled text generation (Hu et al., 2017) framework. The architecture and hyper-parameters of CVAE and controlled CVAE are the same as FVN.

The FVN ablations used in our evaluation are: (1) **FVN-ED** does not use the codebooks, only uses the content and style encoders and decoders, and is equivalent to an attention-augmented sequence-to-sequence model; (2) **FVN-VQ** does not use the content and style encoders and decoders, it directly uses the sampled latent vector for text decoding (3.3); (3) **FVN-EVQ** does not use content and style decoders; (4) **FVN** is the full network. Refer to Table 2 for architecture details. All VAEs and FVN variants are evaluated using multiple references because the sampling from latent space may lead to generate a valid and fluent text that n-gram overlap metrics would not score high when evaluated against a single reference.

4.2 Automatic Evaluation

We evaluate the quality and diversity of the generated text on both dataset. PersonageNLG is style-annotated and delexicalized, so we also report style and content correctness for it.

To evaluate quality in the generated text, we use the automatic evaluation from the E2E generation challenge, which reports BLEU (n-gram precision) (Papineni et al., 2002), NIST (weighted n-gram pre-

Model	BLEU	NIST	METEOR	ROUGE-L
TOKEN†	0.3464	4.9285	0.3648	0.5016
CONTEXT†	0.3766	5.3437	0.3964	0.5255
token-m1†	0.4904	-	-	-
token-m2†	0.4810	-	-	-
token-m3†	0.4906	-	-	-
context-m1†	0.5530	-	-	-
context-m2†	0.5229	-	-	-
context-m3†	0.5598	-	-	-
CVAE	0.9	9.766	0.449	0.702
Controlled CVAE	0.928	9.957	0.463	0.721
FVN-ED	0.802	7.872	0.378	0.696
FVN-VQ	0.887	8.985	0.423	0.715
FVN-EVQ	0.94	**10.129**	0.476	0.748
FVN	**0.965**	9.946	**0.486**	**0.768**

Table 3: Quality Evaluation for PersonageNLG.

Model	Precision	Recall	F_1 score
CVAE	0.961	0.942	0.952
Controlled CVAE	0.961	0.969	0.965
FVN-ED	**0.997**	0.748	0.855
FVN-VQ	0.87	0.799	0.833
FVN-EVQ	0.963	0.989	0.976
FVN	0.987	**1.0**	**0.994**

Table 4: Content Correctness Evaluation for PersonageNLG. Micro precision, recall and F_1 score for "*_SLOT" tokens.

Model	Precision	Recall	F_1 score
CVAE	0.973	0.973	0.973
Controlled CVAE	0.981	0.981	0.981
FVN-ED	0.996	0.996	0.996
FVN-VQ	1.0	1.0	1.0
FVN-EVQ	1.0	1.0	1.0
FVN	**1.0**	**1.0**	**1.0**

Table 5: Style Evaluation on PersonageNLG. Macro precision, recall and F_1 score for the style of generated text based on a separately trained style classifier.

Model	1-gram	2-gram	3-gram	4-gram
ground truth	0.74	0.902	0.924	0.905
CVAE	**0.738**	**0.896**	**0.919**	0.902
Controlled CVAE	0.715	0.869	0.902	0.899
FVN-ED	0.508	0.618	0.668	0.71
FVN-VQ	0.68	0.849	0.896	0.894
FVN-EVQ	**0.738**	0.883	0.907	0.901
FVN	0.720	0.870	0.906	**0.904**

Table 6: Diversity Evaluation on PersonageNLG. Distinct n-grams between generated texts and ground truth.

Model	BLEU	NIST	METEOR	ROUGE-L
TGEN	0.659	8.609	0.448	0.685
SLUG	0.662	8.613	0.445	0.677
Thomson Reuters NLG	0.681	8.778	0.446	0.693
Thomson Reuters NLG	0.674	8.659	0.450	0.698
CVAE	0.377	6.624	0.336	0.525
Controlled CVAE	0.404	6.852	0.346	0.544
FVN-ED	0.665	8.359	0.428	0.699
FVN-VQ	0.681	8.864	0.422	0.698
FVN-EVQ	0.711	**9.066**	**0.453**	**0.721**
FVN	**0.714**	9.004	0.451	0.719

Table 7: Quality Evaluation on E2E.

Model	1-gram	2-gram	3-gram	4-gram
ground truth	0.878	0.949	0.915	0.876
CVAE	0.841	0.931	0.900	0.859
Controlled CVAE	0.834	0.927	0.900	0.859
FVN-ED	0.839	0.924	0.898	0.858
FVN-VQ	**0.855**	**0.943**	0.91	0.869
FVN-EVQ	**0.855**	**0.943**	**0.914**	0.876
FVN	0.841	0.935	0.913	**0.878**

Table 8: Diversity Evaluation on E2E. Distinct n-grams between generated texts and ground truth.

Personality	GT	FVN	p
agreeable	**2.8309**	2.4412	***
conscientiousness	2.9808	**2.9976**	**
disagreeable	2.8345	**2.9388**	***
extravert	2.9221	2.8933	
unconscientiousness	**2.9365**	2.7962	***
overall	**2.9001**	2.8134	***

*:$p < 0.05$, **:$p < 0.01$, ***:$p < 0.001$

Table 9: The analysis result of Question A - grammaticality / naturalness.

cision) (Doddington, 2002), METEOR (n-grams with synonym recall) (Banerjee and Lavie, 2005), and ROUGE (n-gram recall) (Lin, 2004) scores using up to 9-grams. To evaluate content correctness, we report micro precision, recall, and F_1 score of slot special tokens in the generated text, with respect to the slots in the given CMR c. To evaluate diversity, we report the distinct n-grams of ground-truth and baselines' examples. For style evaluation, we separately train a personality classifier (with GloVe embeddings, 3 bi-directional LSTM layers, 2 feed-forward linear layers) on the PersonageNLG training data. The macro precision, recall, and F_1 score of the personality classifier on the test set is 0.996. We use this classifier to evaluate the style of the generated text and report our results in Table 5.

4.3 PersonageNLG Human Evaluation

In addition to automatic evaluation, we conducted a crowdsourced evaluation to compare our model against the ground truth on the entire test set. We did not compare our model with baselines since a pilot evaluation on a random sample of 100 data points from the test set suggested that baselines did not produce fluent enough text to compare with FVN. We considered the ground truth to be a performance upper bound and compared against it to find how close FVN is to it. Crowdworkers were pre-

Personality	GT	equal	FVN	equal or FVN
agreeable	24.46	22.30	**53.24**	**75.54**
conscientiousness	23.38	9.71	**66.91**	**76.62**
disagreeable	**61.87**	12.95	25.18	38.13
extravert	**70.14**	9.35	20.50	29.86
unconscientiousness	**68.71**	4.32	26.98	31.29
overall	**49.71**	11.73	38.56	**50.29**

Table 10: The analysis result of Question B - personality. The percentage frequency distribution (%) over three possible answers (GT, equal, FVN) for each personality is reported, with an additional column reporting the sum of equal and FVN. In this column, underlined values are those that exceed the ones reported in the GT column.

sented with a personality and two sentences (one is ground truth and the other one was generated by FVN) in random order, and were asked evaluate A) the fluency of the sentences in a scale from 1 to 3 and B) which of the two sentences was most likely to be uttered by a person with a given personality (more details in Appendix C). This evaluation was conducted on the entire test set consisting of 1,390 data points, 278 per personality, and each data point was judged by three different crowdworkers.

We report the result of Question A in Table 9. For each sentence, we averaged the scores across three judges. The overall performance of FVN is very close to the ground truth (2.81 vs. 2.9), which suggests that FVN can generate text of comparable fluency with respect to ground truth texts.

We evaluated Question B using a majority vote of the three crowdworkers. Considering the overall performance, 50.29% of times human evaluators considered FVN generated text equal or better at conveying personality than the ground truth. This suggests that FVN can generate text with comparable conveyance with respect to ground truth.

More details and a full breakdown on the human evaluation are available in Appendix C.

4.4 Results and Analysis

Tables 3, 4 and 5 show the results on text quality, content correctness, and style. As shown in Table 3, FVN significantly outperforms the state-of-the-art methods (context-m), especially on BLEU and NIST, which evaluate the precision of generated text, with the caveat regarding single or multiple references explained above. We believe this is due to the fact that FVN explicitly models CMR and style, while context-m depends on human-engineered features. Comparing FVN with CVAE and controlled CVAE, which are similar methods that also sample from the latent space, FVN performs better on all the metrics. Human evaluation results in Section 4.3 show that FVN is close to the ground truth in fluency and style.

Regarding the content correctness evaluation in Table 4, FVN overall performs much better than other baselines, especially on the recall score. Methods with explicit control decoders (controlled CVAE and FVN) perform better than CVAE and FVN-EVQ, which suggests that the controlling module is useful to enhance the content conveyance. Regarding the style evaluation in Table 5, all methods have good performance. Style is likely easy to convey in the text (the markers are pretty specific) and easy to identify for the separately trained personality classifier. Nevertheless, FVN is the best performing model. The text diversity comparison in Table 8 shows how FVN and its ablations have a diversity of generated texts with respect to the ground truth texts, but so do VAE-based methods. The combination of these findings suggests that FVN can produce text with comparable or better diversity than VAEs and ground truth, while conveying content and style more accurately.

Comparing with the ablations, the full FVN always performs better than FVN-ED and FVN-VQ, especially on the recall of slot tokens. FVN-VQ is able to precisely generate slot tokens from the CMR, but it cannot generate all required slot tokens, while FVN can generate them with high precision and substantially higher recall. An explanation is that the latent vectors in the content codebook only memorize the representations of texts without generalizing properly to new CMRs: since FVN is able to generate text containing most of the required slots, that text is usually longer than FVN-VQ's, which also explains why FVN performs better than FVN-VQ on METEOR and ROUGE-L that evaluate the recall of n-grams, and suggests that all encoders and codebooks are indeed needed for obtaining high performance.

The comparison between FVN and FVN-EVQ shows how in some cases FVN-EVQ has higher quality, but FVN obtains better scores on correctness and style, suggesting the additional decoder improves conveyance sacrificing some fluency.

In Table 7, we compare our proposed model and variants against the best performing models in the E2E challenge: TGEN (Novikova et al., 2017), SLUG (Juraska et al., 2018), and Thomson Reuters NLG (Davoodi et al., 2018; Smiley et al., 2018).

extravert	Name EatType Food PriceRange CustomerRating Area FamilyFriendly Near
same e_k^C	let 's see what we can find on Name_SLOT . yeah, it is FamilyFriendly_SLOT with a CustomerRating_SLOT rating , it is a EatType_SLOT , it is a Food_SLOT place in Area_SLOT , it is pricerange_SLOT near Near_SLOT .
different e_n^S	i do n't know . Name_SLOT is a EatType_SLOT with a CustomerRating_SLOT rating , also it is a FamilyFriendly_SLOT , Area_SLOT , and it is a Food_SLOT place near Near_SLOT , also it has a price range of pricerange_SLOT .
	Name_SLOT is a EatType_SLOT , it is a FamilyFriendly_SLOT , it 's a Food_SLOT place , it is near Near_SLOT , it has a CustomerRating_SLOT rating , you know pal! it is in Area_SLOT and has a price range of pricerange_SLOT .
different e_k^C	Name_SLOT is a EatType_SLOT with a CustomerRating_SLOT rating , also it is a Food_SLOT place , you know ! and it is Area_SLOT , also it is FamilyFriendly_SLOT near Near_SLOT , also it has a price range of pricerange_SLOT .
same e_n^S	Name_SLOT is a EatType_SLOT , it is a FamilyFriendly_SLOT , it 's a Food_SLOT place , it is near Near_SLOT , it has a CustomerRating_SLOT rating , you know and it is in Area_SLOT and pricerange_SLOT .
	Name_SLOT is a EatType_SLOT , it is a Food_SLOT place , it is FamilyFriendly_SLOT , it 's in Area_SLOT , it is near Near_SLOT , it has a CustomerRating_SLOT rating and a price range of pricerange_SLOT, you know! .

Table 11: Diversity in FVN-generated PersonageNLG examples. Given the CMR and style the the generated text varies depending on the vector sampled from the codebook.

agreeable	"let 's see what we can find on" "well , i see" "did you say ?" "i suppose" "right" "okay ?" " you see ?" "it is somewhat"
disagreeable	"oh god i mean , everybody knows" "oh god" "i do n't know ." "i am not sure ."
conscientious	"let 's see what we can find on" "well , i see" "did you say " " sort of " "you see ?" "let 's see, " "..."
unconscientious	"oh god i , i do n't know ." "darn" " i mean ." "i ... i , i do n't know ." "i mean , i am not sure ." "damn" "!" "it has like a "
extravert	"oh god i am not sure ." "let 's see ," "..." "alright ?" "yeah" "i do n't know" "did you say ?" "you know !" "you know and" "pal" "! "

Table 12: Top codes' linguistic pattern of each style

We can see from the results that FVN performs better than all these state-of-the-art models. The reason of the low performance of CVAE-based methods on the E2E dataset is that the CMR are disjoint in the train and test sets (while in PersonageNLG they are overlapping) and CVAEs struggle to handle unseen CMRs. FVNs performs well because it builds focused variations for each attribute independently instead of the entire CMR.

Table 11 shows texts generated by FVN under the same CMR (given 8 attributes, rare in training data) and extravert style. The first three samples have the same CMR latent vector, but different sampled style latent vectors. The remaining three examples have different sampled CMR latent vectors, but the same style latent vector. In the first three examples, the generated texts and the words representing the extravert style are different ("let 's see what we can", "I don't know", "you know"). In the latter three examples, the words representing style are similar ("you know"), but the aggregation of attributes is different. These examples suggest that the two codebooks learn disjoint information and that the sampling mechanism introduces the desired variation in the generated texts. Table 11 shows that FVN learns disjoint content and style codebooks and that the vectors in the codebook can be explicitly interpreted by sampling multiple texts and observing the generated patterns. This is useful because, beyond sampling correct style vectors, we can select the realization of a style we prefer(Table 12 shows linguistic patterns associated with the top codes of each style). These patterns are automatically learnt and suggest that there is no need to encode them with manual features. Conditional VAEs do not provide this capability.

Samples obtained providing the same CMR and style to different models and examples of the linguistic patterns learned by FVN's style codebook are provided in Appendix D. Diverse samples obtained from FVN by sampling different latent codes are shown in Table 15.

5 Conclusion

In this paper, we studied the task of controlling language generation, with a specific emphasis on content conveyance and style variation. We introduced FVN, a novel model that overcomes the limitations of previous models, namely lack of conveyance and lack of diversity of the generated text, by adopting disjoint discrete latent spaces for each of the desired attributes. Our experimental results show that FVN achieves state-of-the-art performance on PersonageNLG and E2E datasets and generated texts are comparable to ground truth ones according to human evaluators.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations, San Diego, California, USA*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Douglas Biber. 1991. *Variation across speech and writing*. Cambridge University Press.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. O'Reilly Media, Inc.

Samuel 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.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. *CoRR*, abs/1905.05621.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric C. Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. *ArXiv*, abs/1912.02164.

Elnaz Davoodi, Charese Smiley, Dezhao Song, and Frank Schilder. 2018. The e2e nlg challenge: Training a sequence-to-sequence approach for meaning representation to natural language sentences. In *in prep. for INLG conference*.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

Ondřej Dušek and Filip Jurcicek. 2016a. A context-aware natural language generator for dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 185–190.

Ondřej Dušek and Filip Jurcicek. 2016b. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2018. Findings of the e2e nlg challenge. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 322–328.

Ondřej Dušek, Jekaterina Novikova, and Verena Rieser. 2020. Evaluating the State-of-the-Art of End-to-End Natural Language Generation: The E2E NLG Challenge. *Computer Speech & Language*, 59:123–156.

Jessica Ficler and Yoav Goldberg. 2017. Controlling linguistic style aspects in neural language generation. In *Proceedings of the Workshop on Stylistic Variation*, pages 94–104.

Vrindavan Harrison, Lena Reed, Shereen Oraby, and Marilyn Walker. 2019. Maximizing stylistic control and semantic accuracy in nlg: Personality variation and discourse contrast. *DSNNLG 2019*, page 1.

Ari Holtzman, Jan Buys, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. *ArXiv*, abs/1904.09751.

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596.

Juraj Juraska, Panagiotis Karagiannis, Kevin Bowden, and Marilyn Walker. 2018. A deep ensemble model with slot alignment for sequence-to-sequence natural language generation. 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 152–162.

Juraj Juraska and Marilyn Walker. 2018. Characterizing variation in crowd-sourced data for training neural language generators to produce stylistically varied outputs. In *Proceedings of the 11th International Conference on Natural Language Generation*, pages 441–450.

Urvashi Khandelwal, Omer Levy, Dan Jurafsky, Luke Zettlemoyer, and Mike Lewis. 2020. Generalization through memorization: Nearest neighbor language models. *ArXiv*, abs/1911.00172.

Yuta Kikuchi, Tsutomu Hirao, Hiroya Takamura, Manabu Okumura, and Masaaki Nagata. 2014. Single document summarization based on nested tree structure. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 315–320, Baltimore, Maryland. Association for Computational Linguistics.

Diederik P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations, San Diego, California, USA*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M. Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proc. ACL*.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

François Mairesse and Marilyn Walker. 2007. Personage: Personality generation for dialogue. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 496–503.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The e2e dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206.

Jekaterina Novikova, Oliver Lemon, and Verena Rieser. 2016. Crowd-sourcing nlg data: Pictures elicit better data. In *Proceedings of the 9th International Natural Language Generation conference*, pages 265–273.

Aaron van den Oord, Oriol Vinyals, et al. 2017. Neural discrete representation learning. In *Advances in Neural Information Processing Systems*, pages 6306–6315.

Shereen Oraby, Vrindavan Harrison, Abteen Ebrahimi, and Marilyn Walker. 2019. Curate and generate: A corpus and method for joint control of semantics and style in neural nlg. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5938–5951.

Shereen Oraby, Lena Reed, Shubhangi Tandon, TS Sharath, Stephanie Lukin, and Marilyn Walker. 2018. Controlling personality-based stylistic variation with neural natural language generators. In *Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue*, pages 180–190.

Daniel S Paiva and Roger Evans. 2004. A framework for stylistically controlled generation. In *International Conference on Natural Language Generation*, pages 120–129. Springer.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543. ACL.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Hannah Rashkin, Asli Celikyilmaz, Yejin Choi, and Jianfeng Gao. 2020. Plotmachines: Outline-conditioned generation with dynamic plot state tracking. *arXiv preprint arXiv:2004.14967*.

Hannah Rashkin, Eric Michael Smith, Margaret Li, and Y-Lan Boureau. 2018. Towards empathetic open-domain conversation models: A new benchmark and dataset. *arXiv preprint arXiv:1811.00207*.

Cícero Nogueira dos Santos, Igor Melnyk, and Inkit Padhi. 2018. Fighting offensive language on social media with unsupervised text style transfer. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 2: Short Papers*, pages 189–194. Association for Computational Linguistics.

Xiaoyu Shen, Hui Su, Shuzi Niu, and Vera Demberg. 2018. Improving variational encoder-decoders in dialogue generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. 2019. CTRL: A Conditional Transformer Language Model for Controllable Generation. *arXiv e-prints*, page arXiv:1909.05858.

Charese Smiley, Elnaz Davoodi, Dezhao Song, and Frank Schilder. 2018. The e2e nlg challenge: End-to-end generation through partial template mining. *in prep*.

Kihyuk Sohn, Honglak Lee, and Xinchen Yan. 2015. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pages 3483–3491.

Haoyu Song, Wei-Nan Zhang, Yiming Cui, Dong Wang, and Ting Liu. 2019. Exploiting persona information for diverse generation of conversational responses. *arXiv preprint arXiv:1905.12188*.

Yi-Chia Wang, Runze Wang, Gokhan Tur, and Hugh Williams. 2018. Can you be more polite and positive? infusing social language into task-oriented conversational agents. In *NeurIPS 2018 Workshop on the Second Conversational AI,*.

Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. 2020. Variational template machine for data-to-text generation. *arXiv preprint arXiv:2002.01127*.

Yuchi Zhang, Yongliang Wang, Liping Zhang, Zhiqiang Zhang, and Kun Gai. 2019. Improve diverse text generation by self labeling conditional variational auto encoder. In *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2767–2771. IEEE.

Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.

Dataset	Pairs	Number of Slots in CMR						
		CMRs	3	4	5	6	7	8
Train	88,855	600	0.13	0.30	0.29	0.22	0.06	0.01
Test	1,390	35	0.02	0.04	0.06	0.15	0.35	0.37

Table 13: Distribution of slots in the CMR in both training and test splits of PersonageNLG. Pairs refer to content-utterance pairs.

Dataset	Pairs	CMRs	Number of Slots in CMR					
		CMRs	3	4	5	6	7	8
Train	42061	4862	0.05	0.18	0.32	0.28	0.14	0.03
Dev	4672	547	0.09	0.11	0.05	0.35	0.30	0.10
Test	4693	630	0.01	0.03	0.08	0.17	0.34	0.37

Table 14: Distribution of slots in the CMR in both training, development and test splits of E2E. Pairs refer to content-utterance pairs.

A Experiments details

These are the links to the adopted datasets and the code for computing the metrics.

PersonageNLG text generation dataset: https://nlds.soe.ucsc.edu/stylistic-variation-nlg

End-2-End Challenge dataset (E2E): http://www.macs.hw.ac.uk/InteractionLab/E2E/

Automatic evaluation metrics code from the E2E generation challenge: https://github.com/tuetschek/e2e-metrics

Distinct n-grams metric code: https://github.com/neural-dialogue-metrics

Table 13 shows details of the PersonageNLG dataset, while Table 14 shows details of the E2E dataset.

B Baselines details

The first three baselines are taken from (Oraby et al., 2018) and adopt the TGen architecture (Dušek and Jurcicek, 2016b), an encoder-decoder network, with different kinds of input.

TOKEN adds a token of additional supervision to encode personality. Unlike other works that use a single token to control the generator's output (Hu et al., 2017), the personality token encodes a several different parameters that define style.

CONTEXT introduces a context vector that explicitly encodes a set of 36 manually-defined style parameters encoded as a vector of binary values. We then apply these style encoding approaches to three state of the art models taken from (Harrison et al., 2019), which extend (Oraby et al., 2018) by changing the basic encoder-decoder network to OpenNMT-py (Klein et al., 2017) in the following ways.

m1 inserts style information into the sequence of tokens that constitute the content c;

m2 incorporates style information in the content encoding process by concatenating style representation with content representation before passing it to the content encoder;

m3 incorporates style information into the generation process by additional inputs to the decoder. At each decoding step, style representation is concatenated with each word's embedding and passed as input to the decoder.

token-m means that style (personality here) is encoded with a single token;

context-m means that style is encoded via the 36 parameters.

TGEN (Novikova et al., 2017) adopts a seq2seq model with attention (Bahdanau et al., 2015) with added beam search and a reranker penalizing outputs that stray away from the input CMR.

SLUG (Juraska et al., 2018) adopts seq2seq-based ensemble which uses LSTM/CNN as the encoders and LSTM as the decoder); heuristic slot aligner reranking and data augmentation. Both TGEN and SLUG use partial ('name' and 'near' slot) de-lexicalized texts .

Thomson Reuters NLG (Davoodi et al., 2018; Smiley et al., 2018) use fully de-lexicalized text and a seq2seq model with hyperparameter tunning.

C Human evaluation details

Crowdworkers were presented with a personality and two sentences (one is ground truth and the other one was generated by our model) in random order, and were asked to answer the following two questions:

- Question A: On a scale of 1-3, how grammatical or natural is this sentence? (please answer for both sentences).
- Question B: Which of these two sentences do you think would be more likely to be said by a(n) ___ person? (Fill in ___ with the personality given, e.g. agreeable) Answers: Sentence 1, 2, equally

Question A asked the crowdworkers to assess the degree of grammaticality / naturalness of a sentence while Question B was designed to evaluate which of the two sentences exhibits a specific personality.

area[city centre] customer rating[5 out of 5] eatType[pub] familyFriendly[no] food[French] name[The Phoenix] near[Crowne Plaza Hotel] priceRange[more than £30]
-use the top frequent code of each value-
The Phoenix is a french pub near Crowne Plaza Hotel in the city centre . It is not children friendly and has a price range of more than £30 and has a customer rating of 5 out of 5 .
-use same area[city centre] code, other values' codes are sampled-
The Phoenix is a pub in the city centre . It is a french food . It is located in the city centre .
The Phoenix is a pub in the city centre . It is a french food . It is a high price range and is not child friendly .
-use same customer rating[5 out of 5] code, other values' codes are sampled-
The Phoenix is a french pub located in the city centre . It is a high customer rating and is not children friendly .
The Phoenix is a pub in the city centre near Crowne Plaza Hotel . It is a high customer rating and is not children friendly .
-use same eattype[pub] code, other values' codes are sampled-
The Phoenix is a french pub near Crowne Plaza Hotel in the city centre . It is not children friendly and has a price range of more than £30 .
The Phoenix is a french pub in the city centre near Crowne Plaza Hotel . It is not child friendly and has a high price range and a customer rating of 5 out of 5 .
-use same familyFriendly[no] code, other values' codes are sampled-
The Phoenix is a french pub located in the city centre near Crowne Plaza Hotel . It is not family-friendly and has a customer rating of 5 out of 5 .
The Phoenix is a french pub located in the city centre . It is not family-friendly and has a customer rating of 5 out of 5 .
-use same food[French] code, other values' codes are sampled-
The Phoenix is a french pub in the city centre . It is a high customer rating and is not children friendly .
The Phoenix is a french pub located in the city centre . It is not family-friendly .
-use same priceRange[more than £30] code, other values' codes are sampled-
The Phoenix is a french pub in the city centre . It is not children friendly and has a price range of more than £30 .
The Phoenix is a french pub near Crowne Plaza Hotel in the city centre . It is not children friendly and has a price range of more than £30 .

Table 15: Diversity in FVN-generated E2E examples.

We report the result of Question A in Table 9. For each sentence, we averaged the scores across three judges, and conducted a paired t-test between the ground truth and our model for each personality. The result shows that the FVN sentences were considered significantly more grammatical / natural on conscientiousness and disagreeableness, the ground truth sentences were better on agreeable and unconscientiousness, and no difference was found for extravert. The overall performance of FVN is very close to the ground truth (2.81 vs. 2.9), which suggests that FVN can generate text of comparable fluency with respect to ground truth texts.

We evaluated Question B using a majority vote of the three crowdworkers. Table 10 shows the percentage frequency distribution for each personality and the entire test set. We found that our FVN model performs better than the ground truth on agreeable and conscientiousness, while the ground truth is better for the rest of the three personalities. Specifically, 53% and 67% of the time, the crowdworkers judge the agreeable and conscientious sentences generated by our model to be better than the ground truth sentences. This finding is surprising, since we consider the ground truth be an upper bound in this task, and our model outperforms it two out of five personalities. One possible explanation about why FVN only performs better on agreeable and conscientiousness is that the language patterns of agreeableness and conscientiousness are more systematic and thus easier to learn by the model. In Table 10 we also report a column that shows the percentage frequency of text where the judgment was equal or in favor of FVN. Underlined rows show when the number of equal judgments or judgments favorable to FVN exceeds the judgments that preferred the ground truth text. Considering the overall performance, 50.29% of times human evaluators considered FVN generated text equal or better at conveying personality than the ground truth. This finding suggests that FVN can generate text with comparable conveyance with respect to ground truth texts.

D Generated Samples and Linguistic Patterns

Table 15 shows generated examples from FVN trained on E2E. Given a CMR, we sample a code for each slot value. The first part shows the generated text using the most frequent code for each slot value. We can see that the text is fluent and conveys the CMR precisely. In the remaining part, we keep one slot-value's code fixed and the remaining slot codes are sampled. The fixed slot-value is present, but some of the other slot-values are missing in the generated text. One explanation is that in the training data the text associated with a CMR can also contain missing values and therefore the codebook memorizes this behavior.