Abstract

Natural Language Generation (NLG) for task-oriented dialogue focuses on communicating specific content accurately, fluently, and coherently. While these attributes are crucial for a successful dialogue, it is also desirable to simultaneously accomplish specific stylistic goals, such as response length, point-of-view, descriptiveness, sentiment, formality, and empathy. In this work, we focus on stylistic control and evaluation for schema-guided NLG, with joint goals of achieving both semantic and stylistic control. We experiment in detail with various controlled generation methods for large pretrained language models: specifically, conditional training, guided fine-tuning, and guided decoding. We discuss their advantages and limitations, and evaluate them with a broad range of automatic and human evaluation metrics. Our results show that while high style accuracy and semantic correctness are easier to achieve for more lexically-defined styles with conditional training, stylistic control is also achievable for more semantically complex styles using discriminator-based guided decoding methods. The results also suggest that methods that are more scalable (with less hyper-parameters tuning) and that disentangle content generation and stylistic variations are more effective at achieving semantic correctness and style accuracy.

1 Introduction

Natural Language Generation (NLG) for task-oriented dialogue focuses on effectively generating responses based on inputs that are frequently in the form of a structured meaning representation (MR) (Moryossef et al., 2019; Dušek et al., 2018; Colin et al., 2016; Wen et al., 2015). Recent work has suggested a schema-guided paradigm for task-oriented dialogue by adding descriptions in natural language form (Lin et al., 2021; Du et al., 2020; Rastogi et al., 2019; Bapna et al., 2017). Compared to structured MRs, dialogue schemata contain much richer contextual information, leading to better generated outputs.

Although the primary aim of task-oriented NLG is to effectively generate outputs that realize system dialogue actions and communicate their associated contents correctly, it is often desirable to control the stylistic variations of an output. For example, recognizing and reacting to emotions has been shown to enhance task outcomes and user engagement in task-oriented conversations (Fraser et al., 2018). Language generation systems that use corpora and methods without awareness of emotions may generate callous, generic or even biased responses (Bender et al., 2021; Sheng et al., 2019). Depending on the use case or type of system, it may be useful to stylistically vary responses, e.g., using shorter responses for spoken dialogue systems, longer responses if the system includes visual modality through a screen, or emotion-specific responses that appropriately address user sentiment.

Previous work on controlled text generation aimed at achieving stylistic goals has not focused on a schema-guided paradigm where specific content must be communicated correctly; instead, most work focuses more on unconstrained text-to-text generation without explicit meaning representations (Liu et al., 2021; Krause et al., 2020; Keskar et al., 2019; Ghazvininejad et al., 2017). Meanwhile, work on schema-guided NLG has primarily focused on generating fluent outputs that achieve low semantic error rather than achieving stylistic goals (Kale and Rastogi, 2020; Peng et al., 2020; Du et al., 2020). In this paper, we hope to fill the gap on stylistic control and evaluation for schema-guided NLG. Our contributions in this paper are three-fold:

1. We describe how we pre-process and annotate style parameters within the Schema-guided Dialogue (SGD) dataset (Rastogi et al., 2019).
2. We experiment with controlling different styles with various controlled text generation methods that aim to preserve fluency and semantic correctness.

3. We present results with a broad range of evaluation methods, including a detailed human evaluation.

Specifically, we consider three types of methods: \textit{conditional training}, \textit{guided fine-tuning}, and \textit{guided decoding}. We show that conditional training (CT) can be used for both very lexically-defined (e.g., point-of-view) and more semantically complex styles (e.g., empathy). However, CT introduces the need to re-train new models per style and is more effective at learning styles with strong lexical characteristics (e.g., specific language patterns or vocabulary). For guided fine-tuning, we explore the Plug-and-Play Language Model (PPLM) (Dathathri et al., 2020), but show that it requires careful hyper-parameter tuning and is prone to degeneration. For guided decoding, we evaluate the beam search weighted decoding (BSWD) method and show that it performs best overall on measures of style accuracy for semantically complex styles. The results suggest that unlike style control for unconstrained text generation where no specific content needs to be communicated, style control under the schema-guided paradigm has stronger restrictions on the degree of freedom allowed for content generation. We show that methods that disentangle content generation and style variations, especially for more semantically complex styles, result in better overall performance on semantic and stylistic control.

2 Related Work

Controllable text generation is an emerging research field. Current methods for controlling styles in text generation involve learning a conditional generative model or designing an appropriate decoding strategy. There are many methods proposed for learning a good conditional generative model. These include conditional training (Kikuchi et al., 2016; Ficler and Goldberg, 2017; Keskar et al., 2019; See et al., 2019), fine-tuning language models with external attribute models or side models (Dathathri et al., 2020; Zhang et al., 2020), fine-tuning models with reinforcement learning and human feedback (Ziegler et al., 2019), training generative adversarial models (Yu et al., 2017), and training variational auto-encoders (Yu et al., 2017; Hu et al., 2017).

Others have worked on designing a good decoding strategy to guide generation, where the decoding procedure is guided by a scoring function or discriminator. These include weighted decoding (Ghazvininejad et al., 2017; Holtzman et al., 2018; See et al., 2019) and guided generation (Krause et al., 2020). Other lines of work include curating training data with rich style markup to facilitate training models with explicit stylistic supervision (Oraby et al., 2019, 2018). While this previous work does focus on controllable text generation, most work has been carried out in a text-to-text generation setting, without specific semantic constraints. Instead, we focus on the task-oriented dialogue framework where specific values must be communicated, and conduct a rigorous evaluation of different methods and their efficacy on different forms of style generation. Other recent work has explored adding additional information such as chit-chat data to task-oriented dialogue (Sun et al., 2021; Madotto et al., 2020) and could potentially provide new opportunities for stylistic control.

3 Data Collection and Annotation

3.1 Schema-to-Template pairs

We use the Schema-guided Dialogue (SGD) dataset\footnote{The Schema-guided Dialogue Dataset: \url{https://github.com/google-research-datasets/dstc8-schema-guided-dialogue}} to create a rich corpus of schema-to-template pairs. This dataset is one of the largest publicly available corpora of annotated multi-domain, task-oriented dialogues (Rastogi et al., 2019). Each dialogue in the data is represented as a list of user and system utterances. We use only the system-side utterances and annotations since we are focused on system-side generation.

Table 1 shows an example pre-processed data instance and the final flattened input. To create schema-to-template pairs, we follow the pre-processing steps outlined in Du et al. (2020) with the following modifications: (1) we replace the slot values with generic slot values without any slot type by adding a \$ prefix and appending an increasing index (e.g., \texttt{San Jose} \rightarrow \texttt{$slot1$}) for better generalization; and (2) we use only \textit{domain}, \textit{meaning representation} (MRs), and \textit{slot description} as input data. Domain provides the context of the conversation (e.g., Restaurants); an
MR contains a dialog act, a slot and a value (e.g., 
OFFER(city=$slot2)); and the slot description
describes the meaning of the slot in natural lan-
guage. Table 8 in Appendix A summarizes the full
statistics for the final pre-processed SGD dataset.
In summary, we have 1,698 MRs and 118,715 ex-
ample templates in the training set and 1,137 MRs
and 34,598 templates in the test set.

For a single example, there are usually multiple
dialogue acts as shown in Table 1. An MR in the
original data may also contain multiple values. In
such cases, we flatten these MRs into multiple parts,
each containing only one dialogue act. For
instance, an MR that contains two values origin-
ally, e.g. REQUEST(cuisine=[Mexican, Italian])
becomes two separate dialogue acts, e.g.,
REQUEST(cuisine=$slot1), REQUEST(cuisine=$slot2).
Templates are obtained by delexicalizing the utterances
with generic slot values (e.g., $slot1 is a good
restaurant). Finally, we flatten the input
data into flat natural language strings similar to
Budzianowski and Vulić (2019).

3.2 Style Parameters
In order to perform style control, we also need
to annotate style parameters for the SGD dataset.
Style parameters are features of text that are stylisti-
cally expressive. These parameters can be roughly
identified at lexical (vocabulary and words), syn-
tactic (sentence structure) and semantic (abstract
meaning/emotion) levels (Verma and Srinivasan,
2019). We focus primarily on lexical and semantic
features. Specifically, we characterize lexical style
parameters as low-level linguistic features that can
be derived from the text directly such as word count
and number of adjectives, and semantic style pa-
rameters as high-level styles such as sentiment
that are more complex to characterize.

Lexical style parameters Table 2 summarizes
the lexical styles we annotate for the SGD data
and the description of each parameter. In total, we
automatically annotate six lexical style parameters
for the SGD data. Similar to Zhang et al. (2018),
the parameter “has rare word” uses the maximum
Normalized Inverse Document Frequency (NIDF)
to determine whether or not a template contains
words that are used less frequently in the corpus. The
complete data distribution for all the lexical
parameters is included in Table 10 in Appendix A.

Semantic style parameters Unlike lexical pa-
rameters which consider explicit features such as
vocabulary, semantic parameters of style are less
lexically-defined. As a result, it is generally harder
to annotate these parameters directly from the origi-
nal data without auxiliary information. In this
work, we consider the following semantic param-
eters: formality, negative sentiment, positive sen-
timent and empathy. Formality and sentiment are
common stylistic parameters studied in the stylistic
control NLG literature. We also include empathy as
an interesting and complex style studied in recent
work (Wang et al., 2021; Majumder et al., 2020;
Lin et al., 2019; Zhou and Wang, 2018). We train
a classifier for each of the four styles and annotate
the utterances in SGD with these features. We in-
clude more details about the classifiers in Section 4
and show additional information about the dataset
used to train each classifier in Table 9 of Appendix
A.

4 Baseline Model

Language model Given a sequence of tokens
x = {x_1, , x_T}, the goal of the language model
is to model the joint probability of the sequence
p(x) = p(x_1, , x_T). The joint probability p(x)
is often factorized in terms of the product of condi-
tional probabilities using the chain rule of proba-
bility P(x) = \prod_{t=1}^{T} P(x_t|x_0, , x_{t-1}) (Bengio
et al., 2001). In recent years, transformer-based
models have been used widely to model these con-
ditional probabilities (Vaswani et al., 2017). In
this work, we use GPT-2\footnote{GPT-2 small from HuggingFace: https://huggingface.co/transformers/model_doc/gpt2.html} as our baseline language
model (Radford et al., 2019) and fine-tune the GPT-
2 model with the processed SGD data using a flat
representation with the beginning of sequence, se-
parator, and end of sequence special tokens.\footnote{e.g., “[BOS] flattened-schema-tokens [SEP] template-
tokens [EOS]”}

Semantic style classifiers The classifiers used to
annotate semantic parameters are single layer clas-
sifiers. To train each classifier, we encode the input
x = {x_1, , x_T} of length t using the baseline
GPT-2 model described above and obtain the last
hidden layer o_t for all time steps t. We then take
the average representation across time, denoted o, and
train the classifier to predict the target label (e.g., formal vs. informal) from the average repre-

\[
\text{p}^{(x)}(\text{target label}) = \sum_{t=1}^{T} \text{p}^{(x)}(\text{target label} | \text{input sequence})
\]

This approach allows us to fully leverage the
encoding power of the GPT-2 model for both
lexical and semantic styles.
Table 1: Sample system-side schema and the flat natural language strings.

<table>
<thead>
<tr>
<th>Service</th>
<th>Pre-processed schema</th>
</tr>
</thead>
<tbody>
<tr>
<td>Domain</td>
<td>Restaurants</td>
</tr>
<tr>
<td>Act</td>
<td>OFFER</td>
</tr>
<tr>
<td>Slot</td>
<td>restaurant_name</td>
</tr>
<tr>
<td>Slot Description</td>
<td>Name of the restaurant</td>
</tr>
<tr>
<td>Values</td>
<td>71 Saint Peter</td>
</tr>
<tr>
<td>Act</td>
<td>OFFER</td>
</tr>
<tr>
<td>Slot</td>
<td>City</td>
</tr>
<tr>
<td>Slot Description</td>
<td>City where the restaurant is located</td>
</tr>
<tr>
<td>Values</td>
<td>San Jose</td>
</tr>
<tr>
<td>Utterance</td>
<td>I see that at 71 Saint Peter there is a good restaurant which is in San Jose.</td>
</tr>
<tr>
<td>Template</td>
<td>I see that at $slot1$ there is a good restaurant which is in $slot2$.</td>
</tr>
</tbody>
</table>

Flattened input after pre-processing: restaurants offer restaurant_name name of the restaurant $slot1$ offer city city where the restaurant is located $slot2$

<table>
<thead>
<tr>
<th>Style Parameter</th>
<th>Description</th>
<th>Condition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Short</td>
<td>Number of tokens in the template. The condition is based roughly on 25th percentile of the number of tokens per template in the training data.</td>
<td>$\leq 7$</td>
</tr>
<tr>
<td>Long</td>
<td>Number of tokens in the template. The condition is based roughly on 75th percentile of the number of tokens per template in the training data.</td>
<td>$\geq 15$</td>
</tr>
<tr>
<td>Has Rare Word</td>
<td>Whether or not the template contains very specific or rarely used words compared to other templates in the corpus</td>
<td>Max NIDF $\geq 0.5$</td>
</tr>
<tr>
<td>First Person Pronouns</td>
<td>Whether or not the template contains first person pronouns: I, me, my, mine</td>
<td>-</td>
</tr>
<tr>
<td>Second Person Pronouns</td>
<td>Whether or not the template contains second person pronouns: you, your, yours</td>
<td>-</td>
</tr>
<tr>
<td>Descriptive</td>
<td>Number of adjectives in the template</td>
<td>$&gt; 2$</td>
</tr>
</tbody>
</table>

5 Style Controlled Text Generation

In this work, we require a controlled generation method that is able to simultaneously render the semantic content given the schemata while achieving the desired stylistic goals. We also require the method to be both stable (preserve the fluency of the response even when the stylistic goals are not met) and general-purpose (can be applied to many styles). Under these requirements and constraints, we discuss three types of controlled generation methods to achieve these goals: **conditional training**, **guided fine-tuning**, and **guided decoding**, and compare their performance in Section 6. To our knowledge, our work is the first to systematically study the effectiveness of these control methods for schema-guided NLG.

5.1 Conditional Training (CT)

Controllable generation entails modeling $p(x|a)$, where $a$ is a control variable and $x$ is the generated sequence. However, a pre-trained language model such as GPT-2 is only trained to learn $p(x)$. On the other hand, conditional training (Kikuchi et al., 2016; Peng et al., 2018; Fan et al., 2018; See et al., 2019) refers to directly learning the conditional generative model $p(x|a)$. The results are of high quality because the model is trained to directly maximize $p(x|a)$, but this comes at the expense of fixing the control variable upfront and of re-training the entire model for each new control variable.

To perform style control, we fine-tune the baseline GPT-2 with the conditional training method. Specifically, each input in the training set is annotated with the variable $a$ that we wish to control, e.g., the length (short, long) of the input. The value of the control variable $a$ is then added to model vocabulary as a special token (e.g., $[LENGTH_{SHORT}]$) and appended to the meaning representation after the [BOS] special token. The model then learns an embedding for each value of $a$ and learns to generate $x = \{x_1, \cdots, x_t\}$ con-
ditioned on a value of $a$ and the given meaning representation by optimizing cross-entropy loss.

5.2 Guided Fine-tuning

Unlike conditional training that requires fine-tuning an entire GPT-2 model per style, guided fine-tuning refers to methods that require only fine-tuning a smaller set of the parameters while the majority of the base model stays fixed. In this paper, we consider the recent Plug-and-Play Language Model (PPLM) (Dathathri et al., 2020). In guided fine-tuning methods, the conditional probability $p(x|a) \propto p(x)p(a|x)$ is obtained by fine-tuning the base language model (LM) using an auxiliary discriminator that explicitly models $p(a|x)$. In our work, we use the semantic style classifiers described in Section 4 for the discriminator $p(a|x)$ and the GPT-2 model for the base LM.

The major difficulty of the PPLM method is the problem of degeneration – output that is ungrammatical, incoherent, or repetitive. In practice, we observe that PPLM is prone to generating ungrammatical outputs or getting stuck in a repetitious loop if the hyper-parameters are not carefully tuned. We illustrate the effect of hyper-parameters tuning and the degeneration problem in Appendix B.

5.3 Guided Decoding

While conditional training and guided fine-tuning require fine-tuning the base language model, weighted decoding is applied only at decoding time, requiring no change to the base language model. To control the generation, it re-ranks the probability of words based on a scoring function or discriminator.

Weighted Decoding (WD) In weighted decoding (Ghazvininejad et al., 2017), at time step $t$, the distribution of the next token $x_{t+1}$ is re-weighted by a semantic style classifier that models $p(a|x)$. The probability of each possible next word $w$ in the vocabulary given the control variable $a$ is then re-computed as

$$p(w|a) = \text{Softmax}\left( p(w) p(a|w) \right). \quad (1)$$

Here $p(w)$ is the probability of the word $w$ calculated by the base language model as the next token given the generated sequence $\{x_1, \ldots, x_t\}$, and $p(a|w)$ is the probability of the word $w$ associated with the control variable $a$.

Beam search weighted decoding (BSWD) The weighted decoding method described above takes the highest scoring item at each time step. While this approach is effective, it is often non-optimal and can limit the diversity of the generated text. To mitigate this limitation, we can increase the search space at generation time using the beam search algorithm. Given a fixed beam width parameter $B$, the beam search algorithm selects $B$ best alternatives with the highest probability for an input sequence at each time step. Therefore, the original weighted decoding approach described above is a special case of the beam search algorithm with $B = 1$.

Finally, we note that in Eq. 1, the style classifier is only conditioned on the next possible token $w$ but not the entire past sequence, i.e., the next possible token $w$ plus the text that has been generated $\{x_1, \ldots, x_t, w\}$. Empirically, in both WD and BSWD, we observe that maximizing the probability of the desired style by greedily considering only the next generated token, rather than the entire sequence of previously generated tokens, yielded better performance on the SGD data. When the entire sequence representation is used, we find that the re-weighting of the distribution is usually not strong enough to successfully match the desired stylistic goal.

6 Experiments

In this section, we show the experimental results of the methods described in Section 5 for controlling the styles described in Section 3 on the SGD data. The baseline GPT-2 model is fine-tuned on the training set with no control variables, and the conditional training model is fine-tuned with control variable special tokens, e.g., LENGTH_SHORT. Our evaluation is tested on the test set of 1,137 MRs. We focus on controlling a single style at a time in this experiment; however, it is also possible to control for multiple styles – we include details on multiple-style control experiments in Appendix C (with sample outputs in Appendix Table 14).

6.1 Automatic Evaluation Metrics

We focus on evaluating three key dimensions: style accuracy, fluency, and semantic correctness.

**Style accuracy** To evaluate how effective each controlled generation method is per style, we use the *style accuracy* metric, or the percentage of outputs that conform to the required input style. For lexical styles, this is simply computed using the conditions in Table 2. For semantic styles, we
classify the generated text using the corresponding style classifier and check if the predicted style matches the desired style value. For instance, if the predicted sentiment for generated text with the “positive sentiment” control code does not match the “positive” label, then it is considered incorrect.

Response fluency We use BLEU score (n-gram precision with brevity penalty) (Papineni et al., 2002) as a measurement of the response fluency. We acknowledge that lexical overlap metrics are poor measures of quality (Novikova et al., 2017); however, we include BLEU for completeness and further evaluate quality through human judgments.

Semantic correctness We use slot error rate (SER) (Luong et al., 2015) to measure the semantic correctness of the generated response as compared to the given MR. SER measures the ratio of semantic errors that the model makes by finding the total number of slot mistakes (deletions, repetitions, and hallucinations) in the generated text (lower is better). SER here only considers slots that have explicit values that must be realized (e.g., $slotN$).

6.2 Lexical Style Automatic Evaluation

We evaluate lexical styles with only conditional training as the rest of the methods include semantic style classifiers and are thus not applicable. Table 3 summarizes the performance of conditional training for the six lexical styles. The style accuracy for most styles is generally high, between 80% to nearly 100%, especially for styles marked explicitly by specific words, such as first and second person pronouns. However, we observe that “descriptive” has a particularly low accuracy. First, the majority of the references in the training data (95%) have less than two adjectives, making it difficult for the model to learn this kind of style effectively. Secondly, we observe that conditional training is particularly effective when the style exhibits a clear syntactic pattern or vocabulary; however, this is not the case for the “descriptive” style.

The fluency of the generated text with style control drops slightly as compared to no style control. Having style control often makes the generated text different from its matching responses, i.e., adding extra content or changing linguistic characteristics. Since BLEU score only considers lexical overlap, this behavior is expected. Finally, we see that there is not much of a performance drop in semantic correctness with respect to SER. The experimental results show that for lexical styles with a clear syntactic pattern or vocabulary, CT can be quite effective.

Table 4 illustrates example outputs using CT when controlling for “short”, “long” and “has rare word” styles.6 Interestingly, we see that when asked for a longer response, the model starts to hallucinate extra content (but not slots) not given in the MR in order to satisfy the control variable. This also translates to slightly lower BLEU scores and a higher SER. Methods to enforce better implicit constraints to increase fluency and semantic correctness are important points for future work.

<table>
<thead>
<tr>
<th>Style</th>
<th>Style Acc.</th>
<th>BLEU ↑</th>
<th>SER ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (no style control)</td>
<td>-</td>
<td>0.480</td>
<td>0.010</td>
</tr>
<tr>
<td>Short</td>
<td>96.4%</td>
<td>0.205</td>
<td>0.009</td>
</tr>
<tr>
<td>Long</td>
<td>80.1%</td>
<td>0.381</td>
<td>0.012</td>
</tr>
<tr>
<td>Has Rare Word</td>
<td>89.5%</td>
<td>0.377</td>
<td>0.010</td>
</tr>
<tr>
<td>First Person Pronouns</td>
<td>99.9%</td>
<td>0.365</td>
<td>0.011</td>
</tr>
<tr>
<td>Second Person Pronouns</td>
<td>99.9%</td>
<td>0.447</td>
<td>0.011</td>
</tr>
<tr>
<td>Descriptive</td>
<td>19.6%</td>
<td>0.378</td>
<td>0.012</td>
</tr>
</tbody>
</table>

Table 3: Evaluation of lexical style parameters with conditional training.

6.3 Semantic Style Automatic Evaluation

Table 5 summarizes the main results for semantic style evaluation using CT, PPLM, and BSWD. Since CT is trained to directly maximize conditional probability, it frequently has a higher BLEU score and a lower SER across different styles with the exception of “formal” BLEU. We note that formality is rather lexically-defined, exhibiting characteristic keywords such as “please”, which are frequently picked up by the model, resulting in a particularly high style accuracy for “formal” responses and a lower BLEU score.

For the three more semantically complex styles, “positive sentiment”, “negative sentiment” and “empathy”, we see that BSWD achieves a higher style accuracy than CT (at the cost of a lower BLEU and slightly higher semantic error). We note, however, that the drop in BLEU is expected: as the output is steered towards a certain style, its n-gram overlap with the references is more likely to decrease

5The full data distribution can be found in Appendix A Table 10.

6Example outputs for other lexical styles are included in Appendix C Table 12.
Table 4: Example outputs for conditional training with lexical styles “short”, “long” and “has rare word” (more content hallucinations for “long”). Example outputs for other lexical styles are included in Appendix C Table 12.

– thus, we use these automatic metrics as an evaluation guide, but leave a more rigorous evaluation to human judgment in the next section. Finally, with careful hyper-parameter tuning, PPLM can achieve similar performance to BSWD on BLEU and SER but does worse on style accuracy. Increasing the style accuracy for PPLM worsens the BLEU and SER score significantly and thus we do not consider it in our human evaluation. In summary, our automatic evaluation shows that for semantic styles, BSWD gives us a good trade-off between consistent style accuracy and semantic fidelity across styles.

Table 6 illustrates example outputs for the three semantic styles using BSWD (with additional examples including combining multiple styles in Appendix C). In general, styles that encapsulate complex phenomena such as “empathy” are harder to generate as shown by their lower style accuracy; nevertheless, we are able to preserve fluency and semantic correctness in most cases.

<table>
<thead>
<tr>
<th>Style</th>
<th>Model</th>
<th>Style Acc.</th>
<th>BLEU ↑</th>
<th>SER ↓</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>-</td>
<td>-</td>
<td>0.480</td>
<td>0.010</td>
</tr>
<tr>
<td>Formal</td>
<td>CT</td>
<td>70.0%</td>
<td>0.461</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>PPLM</td>
<td>17.5%</td>
<td>0.390</td>
<td>0.016</td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>42.5%</td>
<td>0.496</td>
<td>0.021</td>
</tr>
<tr>
<td>Negative</td>
<td>CT</td>
<td>7.5%</td>
<td>0.409</td>
<td>0.012</td>
</tr>
<tr>
<td></td>
<td>PPLM</td>
<td>2.5%</td>
<td>0.373</td>
<td>0.017</td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>25.0%</td>
<td>0.359</td>
<td>0.021</td>
</tr>
<tr>
<td>Positive</td>
<td>CT</td>
<td>17.5%</td>
<td>0.442</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>PPLM</td>
<td>32.5%</td>
<td>0.371</td>
<td>0.026</td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>65.0%</td>
<td>0.369</td>
<td>0.021</td>
</tr>
<tr>
<td>Empathy</td>
<td>CT</td>
<td>21.5%</td>
<td>0.449</td>
<td>0.009</td>
</tr>
<tr>
<td></td>
<td>PPLM</td>
<td>11.0%</td>
<td>0.368</td>
<td>0.021</td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>31.0%</td>
<td>0.394</td>
<td>0.017</td>
</tr>
</tbody>
</table>

Table 5: Evaluation of semantic styles.

6.4 Human Evaluation

We focus on human evaluation for our semantic styles since they are the most inherently subjective.\(^7\) We pick a subset of our semantic styles to evaluate, specifically, formal, negative and positive. We also focus on the evaluation of CT and BSWD only since they have an overall better performance in the automatic evaluation and are simpler in nature (e.g., less hyper-parameter tuning). To evaluate style, we ask three human annotators to rate:

- **Style Rating (Sty. Rat.):** How closely the response matches the given style (1 being not at all, 5 being very closely).
- **Fluency (Flu.):** The fluency of the generated response (1 being low, 5 being high).
- **Semantic Error:** Slot errors in the generated response using a 0/1 scale. For analysis purposes, we further break down the items marked “0” into four error types to understand the strengths and weaknesses of each method. For each type of error, the output is only marked “1” if there are no issues.
  - **Deletion (Del.):** Whether the response drops any slot values from the MR.
  - **Repetition (Rep.):** Whether the response repeats any slot values from the MR.
  - **Content Hallucination (Cont. Hal.):** Whether the response includes extra content not given in the MR.
  - **Incorrect Slot Values (Inc. Slot):** Whether the response includes any slot values not given in the MR (a specific type of hallucination).

Table 7 shows the aggregated evaluation (40 samples per style, with three judgments per sample). The results show that outputs generated by BSWD have a higher or comparable style rating.

\(^7\) More details on the human evaluation design are in Appendix D.
compared to CT, confirming its ability to control semantic styles. We also observe that outputs from BSWD have a higher fluency score across all styles. This means that even though BSWD showed lower BLEU scores in automatic evaluation, its outputs are considered to be more natural and fluent in human evaluation. Finally, we see an interesting difference in error types when comparing CT and BSWD. In general, BSWD is more prone to deleting and hallucinating slots, while CT more frequently generates incorrect slot values. Since BSWD requires no change to the base language model, it is able to obtain a lower incorrect slot value error rate as compared to CT, which requires re-training the language model with a control code. On the other hand, it has a higher deletion and hallucination error rate since during decoding time, BSWD is free to insert or drop content in order to achieve the desired style.

Table 6: Example outputs of semantic styles using beam search weighted decoding (BSWD).

<table>
<thead>
<tr>
<th>MR: REQUEST(where_to=none)</th>
<th>W/o style control</th>
<th>What city are you staying?</th>
<th>Formal</th>
<th>What city are you planning to stay in?</th>
</tr>
</thead>
<tbody>
<tr>
<td>MR: OFFER(address=$slot1), OFFER(rating=$slot2)</td>
<td>W/o style control</td>
<td>There is a house at $slot1 with a rating of $slot2.</td>
<td>Positive</td>
<td>There is a nice house with a $slot2 rating located at $slot1.</td>
</tr>
<tr>
<td>MR: INFORM(rating=$slot1), NOTIFY_FAILURE(null=none)</td>
<td>W/o style control</td>
<td>The rating is $slot1. I was unable to make a reservation.</td>
<td>Empathy</td>
<td>The rating is $slot1. I’m sorry, but I couldn’t make the reservation.</td>
</tr>
</tbody>
</table>

Table 7: Human evaluation results for selected semantic styles and methods.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal</td>
<td>CT</td>
<td>3.88</td>
<td>3.83</td>
<td>15%</td>
<td>0%</td>
<td>13%</td>
<td>3.75%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>3.95</td>
<td>4.30</td>
<td>19%</td>
<td>0%</td>
<td>15%</td>
<td>1.25%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Negative</td>
<td>CT</td>
<td>2.78</td>
<td>3.54</td>
<td>20%</td>
<td>0%</td>
<td>12.5%</td>
<td>2.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>2.86</td>
<td>3.93</td>
<td>20%</td>
<td>0%</td>
<td>12.5%</td>
<td>5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>CT</td>
<td>3.48</td>
<td>3.64</td>
<td>22.5%</td>
<td>0%</td>
<td>22.5%</td>
<td>12.5%</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>BSWD</td>
<td>3.42</td>
<td>3.84</td>
<td>27.5%</td>
<td>0%</td>
<td>30%</td>
<td>7.5%</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7 Conclusion

In this work, we focus on stylistic control and evaluation of schema-guided NLG. We discuss three different types of methods for style controlled text generation: conditional training (CT), guided fine-tuning (PPLM), and guided decoding (BSWD). We present a rich set of evaluations to quantify each method’s ability to achieve various styles while preserving language fluency and minimizing slot errors. Our analysis shows that, in general, styles that encapsulate abstract ideas are naturally harder to generate (e.g., empathy), and methods that require careful hyper-parameters tuning may run into the problems of instability and degeneration (e.g., PPLM) while under-performing in style accuracy. The automatic and human evaluations suggest that simultaneously achieving stylistic goals and realizing schema information requires methods that allow us to separate content generation and stylistic variations. We show that CT and BSWD overcome some of these challenges and are effective at controlling several styles while maintaining good fluency and semantic correctness in most cases. CT is effective for lexical styles with strong syntactic characteristics or distinctive vocabulary sets, while BSWD excels at semantic styles that are more complex to characterize.

For future work, we are interested in extending our analysis to a larger number of styles, and in exploring techniques for understanding and representing styles that are more abstract, such as “empathy”, as well as methods for generating those styles under a task-oriented dialogue framework. Additionally, since abstract styles may be characterized by multiple features (e.g., a combination of sentiment and descriptiveness), we are interested in studying how these underlying features can be represented and incorporated more accurately to improve overall semantic and stylistic control.

Acknowledgments

We would like to thank Sofia Scharfenberg, Jasmin Rehm, and the rest of the Alexa Data Services team for all of their help with preparing and performing the human evaluation study.
References


Zhihong Hu, Zichao Yang, Xiaodan Liang, R. Salakhutdinov, and E. Xing. 2017. Toward controlled generation of text. In ICML.


A  Statistics of Style Annotation for SGD Data

<table>
<thead>
<tr>
<th>Style parameters</th>
<th>Dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>Formal</td>
<td>Formality corpus</td>
</tr>
<tr>
<td>Positive</td>
<td>Stanford sentiment treebank (SST)</td>
</tr>
<tr>
<td>Negative</td>
<td>Empathetic reactions</td>
</tr>
</tbody>
</table>

Table 9: Semantic style parameters with their possible values and associated training datasets.

<table>
<thead>
<tr>
<th>Category</th>
<th>Style</th>
<th>Train</th>
<th>Dev</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Lexical styles</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Short</td>
<td>31.42%</td>
<td>36.96%</td>
<td>32.25%</td>
<td></td>
</tr>
<tr>
<td>Long</td>
<td>26.49%</td>
<td>25.13%</td>
<td>25.44%</td>
<td></td>
</tr>
<tr>
<td>Has rare word</td>
<td>35.83%</td>
<td>37.76%</td>
<td>42.04%</td>
<td></td>
</tr>
<tr>
<td>First person pron.</td>
<td>28.48%</td>
<td>28.88%</td>
<td>28.75%</td>
<td></td>
</tr>
<tr>
<td>Second person pron.</td>
<td>52.89%</td>
<td>52.50%</td>
<td>60.18%</td>
<td></td>
</tr>
<tr>
<td>Descriptive</td>
<td>5.00%</td>
<td>5.32%</td>
<td>3.88%</td>
<td></td>
</tr>
<tr>
<td>Positive</td>
<td>39.70%</td>
<td>42.43%</td>
<td>37.41%</td>
<td></td>
</tr>
</tbody>
</table>

Table 10: Data distribution for all of the style parameters we annotate in SGD.

B  Hyper-Parameters Tuning for PPLM

There are several hyper-parameters in PPLM models. We refer the readers to Dathathri et al. (2020) for the full model details. From Section 5.2, the PPLM model can be written as an optimization problem of the form:

$$\min_{\Delta H_t} L_{CE}\left(f\left(\sum_{t=1}^{T} \tilde{o}_t/T\right)\right) + \lambda D_{KL}(\tilde{o}_t||o_t)$$ (4)

s.t.  
$$\tilde{o}_t = LM(x_t, \tilde{H}_t)$$ (5)

$$\tilde{H}_t = H_t + \Delta H_t$$ (6)

$$o_t = LM(x_t, H_t)$$ (7)

where $\lambda$ is a hyper-parameter that scales the KL divergence and $f(\cdot)$ is the semantic style classifier learned to produce $p(a|x)$. We minimize the cross-entropy loss $L_{CE}(\cdot)$ of the attribute model and the Kullback-Leibler (KL) divergence $D_{KL}(\cdot||\cdot)$ of the language model. For each time step $t$, the GPT-2 model generates a history of weight matrices $H_t$ for all hidden layers. To control the generation, we shift the weight matrices $H_t$ in the direction that accomplishes two goals: 1) minimize cross-entropy (CE) of the attribute $a$ under the conditional attribute model $p(a|x)$ and 2) minimize the KL divergence between itself and the unmodified language model $p(x)$. Let $\Delta H_t$ be the small shift of $H_t$ such that $\tilde{H}_t = H_t + \Delta H_t$ can shift the last

Normalized inverse document frequency calculation  
For a word $w$ in a template $d$, the Inverse Document Frequency (IDF) of $w$ is

$$\text{IDF}(w) = \log \left( \frac{N}{1 + |\{d \in D : w \in d\}|} \right)$$ (2)

where $N$ is the total number of templates in the training data $D$, and $|\{d \in D : w \in d\}|$ is the number of those templates $d$ in the training data that contain $w$. The Normalized IDF (NIDF) is obtained via the min-max normalization

$$\text{NIDF}(w) = \frac{\text{IDF}(w) - \min_{w' \in D} (\text{IDF}(w'))}{\max_{w' \in D} (\text{IDF}(w')) - \min_{w' \in D} (\text{IDF}(w'))}$$ (3)

where $\max_{w' \in D} (\text{IDF}(w'))$ and $\min_{w' \in D} (\text{IDF}(w'))$ are the maximum and minimum IDF value of all the words in the training data. We use the maximum NIDF of all the words in a template to determine whether the template contains rarely used words.

Semantic style parameter annotation details 
The formality corpus$^8$ (Pavlick and Tetreault, 2016), which contains sentence-level formality annotations, is used for training the formality classifier. The Stanford sentiment treebank (SST)$^9$ (Socher et al., 2013) is used for training the sentiment classifier. The SST data consists of 5 classes — very negative, negative, neutral, positive, and very positive. We combine “very negative” and “negative” into one class and “very positive” and “positive” into one class. Finally, the empathetic reactions data$^{10}$ (Buechel et al., 2018) is used for training the empathy classifier. We use only the empathy label from the data.

$^8$http://www.seas.upenn.edu/~nlp/resources/formality-corpus.tgz
$^9$https://nlp.stanford.edu/sentiment/
$^{10}$https://github.com/wwbp/empathic_reactions
where $\alpha$ is the step size and the update can be repeated until it converges or up to a certain time step to obtain the shift $\Delta H_t$. At generation time, a post-norm\_fusion that samples the next token $x_{t+1}$ based on the shifted distribution $\tilde{p}_{t+1}$ and the unmodified distribution $p_{t+1}$ is done to increase the language fluency:

$$x_{t+1} \sim \frac{1}{\beta} \left( \frac{\tilde{p}_{t+1}}{p_{t+1}} \right)^{1-\gamma_{gm}}$$  \hspace{1cm} (9)

$\beta$ is a normalizing factor such that it forms a valid distribution. As $\gamma_{gm} \to 1$ this converges to the distribution from the shifted LM, and as $\gamma_{gm} \to 0$ it converges to the unmodified unconditional LM distribution. We observe that the hyper-parameters $\lambda$, $\alpha$, $\gamma_{gm}$ can be tuned to affect the generation. In practice, we found that increasing $\gamma_{gm}$ led to nonsensical results very quickly as the generated text is no longer conformed to the given meaning representation. A larger step size $\alpha$ also led to unstable and nonsensical results compared to a smaller step size as it moves $H_t$ further away from its original position. In practice, a larger $\lambda$ helps keep the fluency and the semantic of the generated text. Nevertheless, its effect is less influential than $\alpha$ and $\gamma_{gm}$.

Table 11 illustrates the effect of hyper-parameters on the model outputs.

### C Additional Experimental Results

We include additional example outputs for lexical styles in Table 12.

#### Multiple Styles Control

In this section, we demonstrate the possibility of controlling multiple styles with the methods described in Section 5. To control multiple styles with conditional training, we condition the CT model on multiple control variables $P(x|a_1, \ldots, a_n)$. The control variables are concatenated and then added to the input as special tokens after the [BOS] token. To control multiple styles with PPLM, we simply add the cross-entropy loss of each semantic style classifier to the objective function. Extra hyper-parameters can be introduced to control the significance of each style. Similarly, for (beam search) weighted decoding, the distribution is re-weighted by

$$p(w|a_1, \ldots, a_n) = \text{Softmax} \left( p(w) \prod_{i=1}^{n} \lambda_i p(a_i|w) \right)$$  \hspace{1cm} (10)

where $\lambda_i$ are hyper-parameters that allow us to determine the significance of each style $a_i$. Finally,
conditional training and (beam search) weighted
decoding can be used simultaneously by training
a CT model and then applying WD or BSWD on
the trained CT model during decoding time. Example
outputs and the style accuracy of multiple style
control using CT and BSWD are shown in Tables
13 and 14.

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Short</td>
<td>55.4%</td>
<td>Formal</td>
<td>2.0%</td>
</tr>
<tr>
<td>Long</td>
<td>95.2%</td>
<td>Formal</td>
<td>39.1%</td>
</tr>
<tr>
<td>Short</td>
<td>69.2%</td>
<td>Positive</td>
<td>20.3%</td>
</tr>
<tr>
<td>Long</td>
<td>84.4%</td>
<td>Positive</td>
<td>44.8%</td>
</tr>
</tbody>
</table>

Table 13: Style accuracy of multiple-style control using CT and BSWD with beam width $B = 2$. The parameter *length* is controlled by CT and the parameters *formality* and *sentiment* are controlled by the BSWD.

**D Human Evaluation Design**

The goal of the annotation task is to evaluate the
style of the automatically generated templates that
express specific information to the user. The tem-
plates follow a certain style value such as “formal”.
The annotators are asked to imagine that they are
having a conversation with the system, and that
the template presented to them is an example of
something that the system may say to them. The
template may be in the form of a question, state-
ment, confirmation, etc. Table 15 illustrates two
sample MRs and four generated templates, which
we lexicalize with a map of values (one per slot) to
make the annotation task more intuitive.
Table 14: Example outputs by combining CT and BSWD with beam width $B = 2$. Lexical style is controlled by CT and semantic style is controlled by the BSWD. Note how the lexical style "long" tends to yield outputs that include more hallucinated content in an attempt to fulfill the required style goal(s).

Table 15: Example human evaluation worksheet.