Steering Output Style and Topic in Neural Response Generation

Di Wang[1,2], **Nebojsa Jojic**[1], **Chris Brockett**[1], **and Eric Nyberg**[2]
[1]Microsoft Research, Redmond, WA, USA
[2]School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, USA
diwang@cs.cmu.edu, jojic@microsoft.com

Abstract

We propose simple and flexible training and decoding methods for influencing output style and topic in neural encoder-decoder based language generation. This capability is desirable in a variety of applications, including conversational systems, where successful agents need to produce language in a specific style and generate responses steered by a human puppeteer or external knowledge. We decompose the neural generation process into empirically easier sub-problems: a faithfulness model and a decoding method based on selective-sampling. We also describe training and sampling algorithms that bias the generation process with a specific language style restriction, or a topic restriction. Human evaluation results show that our proposed methods are able to to restrict style and topic without degrading output quality in conversational tasks.

1 Introduction

Neural encoder-decoder models have demonstrated great promise in many sequence generation tasks, including neural machine translation (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2014; Luong et al., 2015; Wu et al., 2016), image captioning (Xu et al., 2015), summarization (Rush et al., 2015; Gu et al., 2016; Kikuchi et al., 2016), and conversation generation (Vinyals and Le, 2015; Sordoni et al., 2015; Shang et al., 2015; Shao et al., 2017; Li et al., 2015). These encouraging early successes have motivated research interest in training more natural-sounding conversational systems based on large volumes of open-domain human-to-human interactions. In order to create more human-like patterns of conversation, the agents need to have recognizable (and tunable) style, just as individual humans do, and also need to accept guidance from separate information processing modules in order to increase quality of responses. In an extreme case, an agent may be micro-managed by a human user who uses the neural model to enforce grammar and style (e.g., a level of politeness, or a type of humor), while driving the content directly (e.g., by expressing In this manner, the neural model becomes an authoring tool, rather than an independent chat-bot. On the other hand, in fully automated agent systems, the agent may be influenced by a knowledge database, or some other artificial information system, while running in a pre-set style or a style deemed best based on the course of the conversation.

One obstacle to achieving this with neural language generation models is that the sentence representation is distributed across all coordinates of

chat context	Where are you?
Twitter bot	i'm on my way to london now.
Star Wars bot	we're heading to the dark side of jabba's palace.
Hillary bot	i'm here in philadelphia, pennsylvania
Trump bot	i went to the white house. we were in brooklyn.
Kennedy bot	i am on the question of the united states.
chat context	What are you doing?
Twitter bot	i'm just chilling with my dad, i'm just sitting in the kitchen and listening to music
Star Wars bot	i'm doing my magic solo.
Hillary bot	i'm running for president, i'm going to be talking about some of these things
Trump bot	i'm building a wall, right now
Kennedy bot	today, i am asking the congress for a new program to make a new effort to increase the tax privileges and to stimulate

Table 1: Example responses from our Star Wars, Hillary, Trump, and Kennedy bots with scented conversation models.

the embedding vector in a way that is hard to disentangle, and thus control. In order to gain insight into the full distribution of what a decoder might produce given the prompt sentence as input, the model has to be heavily (and sometimes cleverly) sampled. The second problem is that neural models only become highly functional after training with very large amounts of data, while the strongly recognizable style usually must be defined by a relatively tiny corpus of examples (e.g., all Seinfeld episodes, or all popular song lyrics).

In this paper, we address the challenge of how to enforce the decoder models to mimic a specific language style with only thousands of target sentences, as well as generating specific content in that style. We developed and experimented with several training and decoding procedures to allow the model to adapt to target language style and follow additional content guidance. Our experiments, conducted on an open-domain corpus of Twitter conversations and small persona corpora, show that our methods are capable of responding to queries in a transferred style without significant loss of relevance, and can respond within a specific topic as restricted by a human. Some examples of 'scenting' the base conversation model with particular styles are shown in Table 1. More can be found in the Supplementary Material.

2 Related Work

Recurrent neural network based encoder-decoder models have been applied to machine translation and quickly achieved state-of-the-art results (Bahdanau et al., 2014; Luong et al., 2015). As an extension, the attention mechanism enables the decoder to revisit the input sequence's hidden states and dynamically collects information needed for each decoding step. Specifically, our conversation model is established based on a combination of the models of (Bahdanau et al., 2014) and (Luong et al., 2015) that we found to be effective. In section 3, we describe the attention-based neural encoder-decoder model we used in detail.

This work follows the line of research initiated by (Ritter et al., 2011) and (Vinyals and Le, 2015) who treat generation of conversational dialog as a data-drive statistical machine translation (SMT) problem. Sordoni et al. (2015) extended (Ritter et al., 2011) by re-scoring SMT outputs using a neural encoder-decoder model conditioned on conversation history. Recently, researchers have used neural encoder-decoder models to directly generate responses in an end-to-end fashion without relying on SMT phrase tables(Vinyals and Le, 2015; Sordoni et al., 2015; Shang et al., 2015; Shao et al., 2017; Li et al., 2015).

Li et al. (2016) defined a "persona" as the character that an artificial agent, as actor, plays or performs during conversational interactions. Their dataset requires user identification for all speakers in the training set, while our methods treat the base data (millions of twitter conversations) as unlabeled, and the target persona is defined simply by a relatively small sample of their speech. In this sense, the persona can be any set of text data. In our experiments, for example, we used a generic Star Wars character that was based on the entire set of Star Wars scripts (in addition to 46 million base conversations from Twitter). This provides us with a system that can talk about almost anything, being able to respond to most prompts, but in a recognizable Star Wars style. Other possibilities include training (styling) on famous personalities, or certain types of poetry, or song lyrics, or even mixing styles by providing two or more datasets for styling. Thus our targets are highly recognizable styles, and use of these for emphasis (or caricature) by human puppeteers who can choose from multiple options and guide neural models in a direction they like. We expect that these tools might not only be useful in conversational systems, but could also be popular in social media for text authoring that goes well beyond spelling/grammar auto correction.

3 Neural Encoder-Decoder Background

In general, neural encoder-decoder models aim at generating a target sequence $Y = (y_1, \ldots, y_{T_y})$ given a source sequence $X = (x_1, \ldots, x_{T_x})$. Each word in both source and target sentences, x_t or y_t, belongs to the source vocabulary V_x, and the target vocabulary V_y respectively.

First, an encoder converts the source sequence X into a set of context vectors $C = \{\mathbf{h}_1, \mathbf{h}_2, \ldots, \mathbf{h}_{T_x}\}$, whose size varies with regard to the length of the source passage. This context representation is generated using a multi-layered recurrent neural network (RNN). The encoder RNN reads the source passage from the first token until the last one, where $\mathbf{h}_i = \Psi\left(\mathbf{h}_{i-1}, \mathbf{E}_x\left[x_t\right]\right)$. Here $\mathbf{E}_x \in \mathbb{R}^{|V_x| \times d}$ is an embedding matrix containing vector representations of words, and Ψ is

a recurrent activation unit that we employ in the Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber, 1997).

The decoder, which is also implemented as an RNN, generates one word at a time, based on the context vector set returned by the encoder. The decoder's hidden state $\bar{\mathbf{h}}_t$ is a fixed-length continuous vector that is updated in the same way as encoder. At each time step t in the decoder, a time-dependent attentional context vector $\mathbf{c}_t$ is computed based on the current hidden state of the decoder $\bar{\mathbf{h}}_t$ and the whole context set C.

Decoding starts by computing the content-based score of each context vector as: $e_{t,i} = \bar{\mathbf{h}}_t^\top W_a \mathbf{h}_i$. This relevance score measures how helpful the i-th context vector of the source sequence is in predicting next word based on the decoder's current hidden state $\bar{\mathbf{h}}_t^\top$. Relevance scores are further normalized by the softmax function: $\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{T_x} \exp(e_{t,j})}$, and we call $\alpha_{t,i}$ the attention weight. The time-dependent context vector $\mathbf{c}_t$ is then the weighted sum of the context vectors with their attention weights from above: $\mathbf{c}_t = \sum_{i=1}^{T_x} \alpha_{t,i} \mathbf{h}_i$.

With the context vector $\mathbf{c}_t$ and the hidden state $\mathbf{h}_t$, we then combine the information from both vectors to produce an attentional hidden state as follow: $\mathbf{z}_t = \tanh(\mathbf{W}_c[\mathbf{c}_t; \mathbf{h}_t])$. The probability distribution for the next target symbol is computed by $p(y_t = k|\tilde{y}_{<t}, X) \propto \exp(\mathbf{W}_s \mathbf{z}_t + \mathbf{b}_t)$.

4 Decoding with Selective Sampling

The standard objective function for neural encoder-decoder models is the log-likelihood of target T given source S, which at test time yields the statistical decision problem:

$$\hat{T} = \arg\max_T \{ \log p(T|S) \}. \quad (1)$$

However, as discussed in (Li et al., 2015; Shao et al., 2017), simply conducting beam search over the above objective will tend to generate generic and safe responses that lack diversity, such as *"I am not sure"*. In section 7.3, we present a ranking experiment in which we verify that an RNN-based neural decoder provides a poor approximation of the above conditional probability, and instead biases towards the target language model $p(T)$. Fortunately, the backward model $p(S|T)$ empirically perform much better than $p(T|S)$ on the relevance ranking task. Therefore, we directly apply Bayes' rule to Equation 1, as in statistical machine translation (Brown et al., 1993), and use:

$$\hat{T} = \arg\max_T \{ \log p(S|T) + \log p(T) \}. \quad (2)$$

Since $p(T|S)$ is empirically biased towards $p(T)$, in practice, this objective also resembles the Maximum Mutual Information (MMI) objective function in (Li et al., 2015).

The challenge now is to develop an effective search algorithm for a target words sequence that maximize the product in Equation 2. Here, we follow a similar process as in (Wen et al., 2015) which generates multiple target hypotheses with stochastic sampling based on $p(T|S)$, and then ranks them with the objective function 2 above. However, as also observed by (Shao et al., 2017), step-by-step naive sampling can accumulate errors as the sequence gets longer.

To reduce language errors of stochastic sampling, we introduce a sample selector to choose the next token among N stochastically sampled tokens based on the predicted output word distributions. The sample selector, which is a multilayer perceptron in our experiments, takes the following features: 1) the log-probability of current sample word in $p(w_t|S)$; 2) the entropy of current predicted word distribution, $\sum_{w_t} P(w_t|S) \log P(w_t|S)$ for all w_t in the vocabulary; 3) the log-probability of current sample word in $p(w_t|\emptyset)$, which we found effective in ranking task. The selector outputs a binary variable that indicates whether the current sample should be accepted or rejected.

At test time, if none of the N sampled tokens are above the classification threshold, we choose the highest scored token. If there are more than 1 acceptable samples among N stochastically sampled tokens, we randomly choose one among them. Ideally, this permits us to safely inject diversity while maintaining language fluency. We also use the sample acceptor's probabilities as the language model score $P(T)$ for objective in equation 2.

As regards directly integrating beam-search, we found (a) that beam-search often produces a set of similar top-N candidates, and (b) that decoding with only the objective $p(Y|X)$ can easily lead to irrelevant candidates. (See section 7.3) Therefore, we use the selective-sampling method to generate candidates for all our experiments; this (a) samples stochastically then (b) selects using a learned objective from data. The sample-then-select ap-

proach encourages more diversity (v.s. MMI's beam-search) while still maintain language fluency (v.s. naive-sampling).

5 Output style restriction using a small 'scenting' dataset

In this section, we propose three simple yet effective methods of influencing the language style of the output in the neural encoder-decoder framework. Our language style restricting setup assumes that there is a large open-domain parallel corpus that provides training for context-response relevance, and a smaller monologue speaker corpus that reflects the language characteristics of the target speaker. We will refer to this smaller set as a 'scenting' dataset, since it hints at, or insinuates, the characteristics of the target speaker.

5.1 $Rank$: Search in the Target Corpus

Our first approach to scenting is to simply use the all sentences in the target speaker's corpus as generation candidates, ranked by the objective (2) for a given prompt. Since these sentences are naturally-occurring instead of generated word-by-word, we can safely assume $p(T)$ is constant (and high), and so the objective only requires sorting the sentences based on the backward model $p(S|T)$.

RNN-based ranking methods are among the most effective methods for retrieving relevant responses (Wang and Nyberg, 2015, 2016). Thus this approach is a very strong baseline. Its limitation is also obvious: by limiting all possible responses to a fixed finite set of sentences, this method cannot provide a good response if such a response is not already in the scenting dataset.

5.2 $Multiply$: Mixing the base model and the target language model during generation

In our second method we use both the vanilla encoder-decoder model trained on open-domain corpus and the target domain language model trained on the corpus while decoding output sentence. The idea is to use a speaker's language model, which is also RNN-based in our experiments, to restrict the open-domain encoder-decoder model's step-by-step word prediction. Similar ideas have been tested in domain adaptation for statistical machine translation (Koehn and Schroeder, 2007), where both in-domain and open-domain translation tables were used as candidates for generating target sentence. Because open-domain encoder-decoder models are trained with various kinds of language patterns and topics, choosing a sequence that satisfies both models may produce relevant responses that are also in the target language style. We found that a straightforward way of achieving this is to multiply the two models' distributions $p_1(t|S)^{\lambda_1} p_2(t)^{\lambda_2}$ at each point and then re-normalize before sampling. The weights can be tuned either by the perplexity on the validation set, or through manually controlling the trade-off between style restriction and answer accuracy.

5.3 $Finetune$: Over-training on Target Corpus with Pseudo Context

Fine-tuning is a widely used in the neural network community to achieve transfer learning. This strategy permits us to train the neural encoder-decoder on a larger general parallel corpus, and then use the learned parameters to initialize the training of a styled model. Most of the time, however, the target speaker's corpus will lack training data in parallel form. For example, if we train on song lyrics or movie scripts, or political speeches, the data will not be in a question-answer form. To make encoder-decoder overtraining possible, we treat every sentence in the scenting corpus as a target sentence T generated a pseudo context from the backward model $p(S|T)$ trained on the open-domain corpus. Over-training on such pairs imparts the scenting dataset's language characteristics, while retaining the generality of the original model. We also found that the previous sentence in the styled corpus (i.e., previous sentence in the speech) provides helpful context for the current sentence, analogous with a question-answer link. Thus we use both pseudo context and the previous sentence as possible sources S to fine-tune the in-domain decoder. To avoid overfitting, we stop overtraining when the perplexity on the in-domain validation set starts to increase. A corresponding sample acceptor is also trained for the fine-tuned model: we found it helpful to initialize this from the open-domain model's sample acceptor.

6 Restricting the Output Topic

We further introduce a topic restricting method for neural decoders based on the Counting Grid (Jojic and Perina, 2011) model, by treating language guidance as a topic embedding. Our model exten-

sion provides information about the output topic in the form of an additional topic embedding vector to the neural net at each time step.

6.1 *CG*: Counting Grids

The basic counting grid $\pi_{\mathbf{k}}$ is a set of distributions on the d-dimensional toroidal discrete grid $\mathbf{E}$ indexed by $\mathbf{k}$. The grids in this paper are bi-dimensional and typically from $(E_x = 32) \times (E_y = 32)$ to $(E_x = 64) \times (E_y = 64)$ in size. The index z indexes a particular word in the vocabulary $z = [1 \dots Z]$. Thus, $\pi_{\mathbf{i}}(z)$ is the probability of the word z at the d-dimensional discrete location $\mathbf{i}$, and $\sum_z \pi_{\mathbf{i}}(z) = 1$ at every location on the grid. The model generates bags of words, each represented by a list of words $\mathbf{w} = \{w_n\}_{n=1}^N$ with each word w_n taking an integer value between 1 and Z. The modeling assumption in the basic CG model is that each bag is generated from the distributions in a single window $\mathbf{W}$ of a preset size, e.g., $(W_x = 5) \times (W_y = 5)$. A bag can be generated by first picking a window at a d-dimensional location ℓ, denoted as W_ℓ, then generating each of the N words by sampling a location $\mathbf{k}_n$ for a particular micro-topic $\pi_{\mathbf{k}_n}$ uniformly within the window, and sampling from that micro-topic.

Because the conditional distribution $p(\mathbf{k}_n|\ell)$ is a preset uniform distribution over the grid locations inside the window placed at location ℓ, the variable $\mathbf{k}_n$ can be summed out (Jojic and Perina, 2011), and the generation can directly use the grouped histograms

$$h_\ell(z) = \frac{1}{|\mathbf{W}|} \sum_{\mathbf{j} \in W_\ell} \pi_{\mathbf{j}}(z), \tag{3}$$

where $|\mathbf{W}|$ is the area of the window, e.g. 25 when 5×5 windows are used. In other words, the position of the window ℓ in the grid is a latent variable given which we can write the probability of the bag as

$$P(\mathbf{w}|\ell) = \prod_{w_n \in \mathbf{w}} h_\ell(w_n) = \prod_{w_n \in \mathbf{w}} \Big(\frac{1}{|\mathbf{W}|} \cdot \sum_{\mathbf{j} \in W_\ell} \pi_{\mathbf{j}}(w_n) \Big) \tag{4}$$

As the grid is toroidal, a window can start at any position and there is as many h distributions as there are π distributions. The former will have a considerably higher entropy as they are averages of many π distributions. Although the basic CG model is essentially a simple mixture assuming the existence of a single source (one window) for all the features in one bag, it can have a very large number of (highly related) choices h to choose from. Topic models (Blei et al., 2003; Lafferty and Blei, 2006), on the other hand, are admixtures that capture word co-occurrence statistics by using a much smaller number of topics that can be more freely combined to explain a single document (and this makes it harder to visualize the topics and pinpoint the right combination of topics to use in influencing the output).

In a well-fit CG model, each data point tends to have a rather peaky posterior location distribution because the model is a mixture. The CG model can be learned efficiently using the EM algorithm because the inference of the hidden variables, as well as updates of π and h can be performed using summed area tables (Crow, 1984), and are thus considerably faster than most of the sophisticated sampling procedures used to train other topic models. The use of overlapping windows helps both in controlling the capacity of the model and in organizing topics on the grid automatically: Two overlapping windows have only slighly different h distributions, making CGs especially useful in visualization applications where the grid is shown in terms of the most likely words in the component distributions π (Perina et al., 2014).[1]

Having trained the grid on some corpus (in our case a sample of the base model's corpus), the mapping of either a source S and/or target T sentence can be obtained by treating the sentences as bags of words. By appending one or both of these mappings to the decoder's embedding of the target T, the end-to-end encoder-decoder learning can be performed in a scenario where the decoder is expected to get an additional hint through a CG mapping. In our experiments, we only used the embedding of the target T as the decoder hint, and we appended the full posterior distribution over CG locations to the encoder's embedding. At test time, we only have the S and need to generate T without knowing where it may map in the counting grid. We considered two ways of providing a mapping:

- The user provides a hint sentence H (could be just a few words in any order), and the CG mapping of the user's hint, i.e. the full posterior distribution $p(\ell|H)$, is used in the decoding. The posterior probabilities over 32×32 grid locations are unwrapped into a vector

[1] (Chen et al., 2017) have recently proposed using LDA for topic modeling in Sequence-To-Sequence response generation models. We believe that the CG embedding used here will prove easier to apply and interpret through visualization.

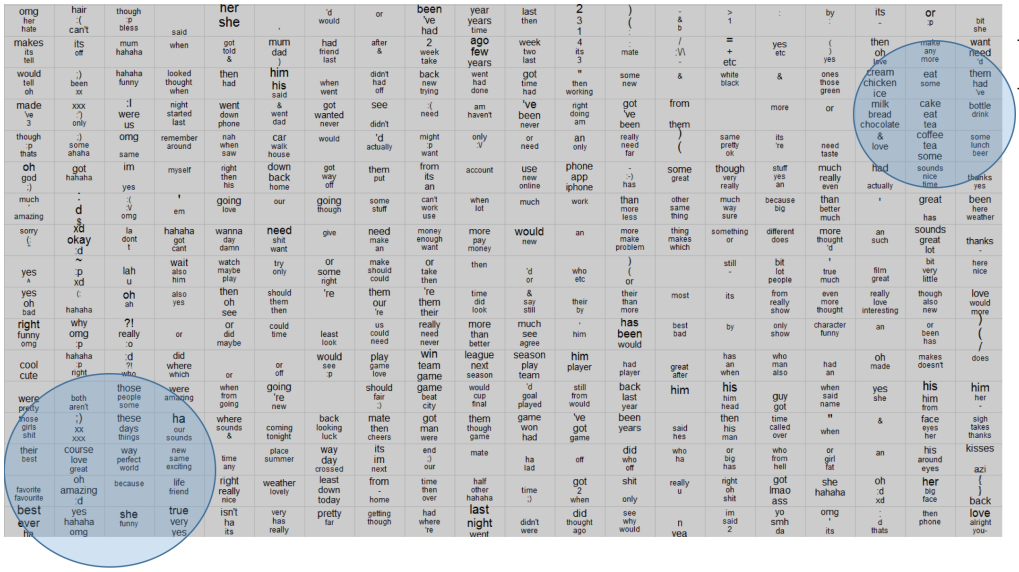

Figure 1: A part of a Counting Grid trained on Twitter data and its use in providing topical hints in decoding. For the source sentence at the top, the decoder may produce the two target samples on the right, if the circled locations are used as a hint, or the two sentences at the bottom if the locations in the lower right are picked.

with a size of $|L| = 1024$, and then concatenated with the word embedding as the input at each time-step. That acts to expand the user's hint into a sentence with similar content (and style if the model is also styled).

- The CG is scanned and a variety of mappings are tested as inputs to provide a diverse set of possible answers. In our experiments, instead of scanning over all 1024 possible locations in the grid, we retrieved several possible answers using information retrieval (ranking of the data samples in the training set based on the source S and picking the top ten). Then the CG mapping $p(\ell|H)$ of these retrieved hints is used to decode several samples from each.

As an example, Figure 1 shows a portion of a CG trained on randomly chosen 800k tweets from the twitter corpus. In each cell of the grid, we show the top words in the distribution $\pi_{\mathbf{j}}(z)$ over words (z) in that location ($\mathbf{j}$). (Each cell has a distribution over the entire vocabulary). As a response to "I am hungry," using two highlighted areas as hints, we can generate either a set of empathic responses, such as 'Me too,' or food suggestions, such as 'Let's have cake.' It will also be evident that some areas of the grid may produce less sensical answers. These can later be pruned by likelihood criteria or by user selection.

7 Experiments

7.1 Datasets

Yahoo! Answer Dataset. We use the Comprehensive Questions and Answers dataset[2] to train and validate the performances of different decoding setups with ranking experiments described in section 7.3. This dataset contains 4.4 million Yahoo! Answers questions and the user-selected best answers. Unlike the conversational datasets, such as the Twitter dataset described below, it contains more relevant and specific responses for each question, which leads to less ambiguity in ranking.

Twitter Conversation Dataset. We trained our base encoder-decoder models on the Twitter Conversation Triple Dataset described in (Sordoni et al., 2015), which consists of 23 million conversational snippets randomly selected from a collection of 129M context-message-response triples extracted from the Twitter Firehose over the 3-month

[2] http://webscope.sandbox.yahoo.com/catalog.php?datatype=l

period from June through August 2012. For the purposes of our experiments, we split the triples into context-message and message-response pairs yielding 46M source-target pairs. For tuning and evaluation, we used the development dataset of size 200K conversation pairs and the test dataset of 5K examples. The corpus is preprocessed using a Twitter specific tokenizer (O'Connor et al., 2010). The vocabulary size is limited to 50,000 excluding the special boundary symbol and the unknown word tag.

Scenting datasets. A variety of persona characters have been trained and tested, including Hillary Clinton, Donald Trump, John F. Kennedy, Richard Nixon, singer-songwriters, stand-up comedians, and a generic Star Wars character. In experiments, we evaluated on a diverse set of representative target speakers:

JFK. We mainly tested our models on John F. Kennedy's speeches collected from American Presidency Project[3], which contains 6474 training and 719 validation sentences.

Star Wars. Movie subtitles of three Star Wars movies are also tested[4]. They are extracted from Cornell Movie-Dialogs Corpus (Danescu-Niculescu-Mizil and Lee, 2011), and have 495 training and 54 validation sentences.

Singer-Songwriter. We also evaluated our approach on a lyric corpus from a collective of singers: Coldplay, Linkin Park, and Green Day. The lyric dataset is collected from mldb.org and has 9182 training and 1020 validation lines.

Debate Chat Contexts. We designed testing questionnaires with 64 chat contexts spanning a range of topics in politic, science, and technology: the sort of questions we might ask in an entertaining political debate.[5] To test the model's ability to control output topic in section 7.4.3, we also created one hint per question.

7.2 Network Setup and Implementation

Our encoder and decoder RNNs contains two-layer stacked LSTMs. Each LSTM layer has a memory size of 500. The network weights are randomly initialized using a uniform distribution $(-0.08, 0.08)$, and are trained with the ADAM optimizer (Kingma and Ba, 2014), with an initial learning rate of 0.002. Gradients were clipped so their norm does not exceed 5. Each mini-batch contains 200 answers and their questions. The words of input sentences were first converted to 300-dimensional vector representations learned from the RNN based language modeling tool word2vec (Mikolov et al., 2013). The beginning and end of each passage are also padded with a special boundary symbol. During decoding, our model generates 500 candidate samples in parallel, then ranks them. As these are processed in batches on GPU, generation is very efficient. We also experimented incorporating an information retrieval (IR) module to automatically collect topic hints for CG-based decoder. Specifically, a full-text index of twitter corpus is built using solr[6], and the top 10 searched results based on the source sentence are be used to generate posterior CG distributions as hints.

7.3 Validating the Decoding Setup with Ranking

We performed a ranking evaluation applying different decoding setups on the Yahoo! Answers dataset. Here we wanted to test the relevance judgment capacities of different setups, and validate the necessity of the new decoding method discussed in section 4. Yahoo! Answers question is used as source S, and its answer is treated as target T. Each test question is associated with one true answer and 19 random answers from the test set. MRR (Mean Reciprocal Rank) and P@1 (precision of top1) were then used as evaluation metrics.

Table 2 shows the answer ranking evaluation results: the forward model $P(T|S)$, by itself is close to the performance of random selection in distinguishing true answer from wrong answers. This implies that a naive beam search over only the forward model may generate irrelevant outputs. One hypothesis was that $P(T|S)$ is biased toward $P(T)$, and performance indeed improves after normalizing by $P(T)$. However, it is difficult to directly decode with objective $P(T|S)/P(T|\emptyset)$, because this objective removes the influence of the target-side language model. Decoding only according to this function will thus result in only low-frequency words and ungrammatical sentences, behavior also noted by (Li et al., 2015; Shao et al., 2017).

[3] http://www.presidency.ucsb.edu/

[4] Koncel-Kedziorski et al. (2016) also uses Star Wars scripts to test theme rewriting of algebra word problems.

[5] See the Supplementary material.

[6] https://lucene.apache.org/solr/

Ranking Methods	MRR	P@1
$P_{rnn}(T\|S)$	0.224	0.075
$P_{rnn}(T\|S)/P_{rnn}(T\|\emptyset)$	0.652	0.524
$P_{rnn}(S\|T)$	0.687	0.556

Table 2: Ranking the true target answer among random answers on Yahoo! Answers test set.

7.4 Human Evaluations

7.4.1 Systems

We tested 10 different system configurations to evaluate the overall output quality, and their abilities of influencing output language style and topic:

- ***vanilla-sampling*** each word in the target.
- ***selective-sampling*** as described in section 4; all the following systems are using it as well.
- ***cg-ir*** uses IR results to create counting grid topic hints (sections 6.1 and 7.2).
- ***rank*** uses proposals from the full JFK corpus as in section 5.1.
- ***multiply*** with a JFK language model as in section 5.2.
- ***finetune*** with JFK dataset as in section 5.3.
- ***finetune-cg-ir*** uses IR results as topic hints for fine-tuned JFK.
- ***finetune-cg-topic*** forced to use the given topic hint for fine-tuned JFK.
- ***singer-songwriter*** fine-tuned cg-topic.
- ***starwars*** fine-tuned cg-topic.

7.4.2 Evaluation Setup

Owing to the low consistency between automatic metrics and human perception on conversational tasks (Liu et al., 2016; Stent et al., 2005) and the lack of true reference responses from persona models, we evaluated the quality of our generated text with a set of judges recruited from Amazon Mechanical Turk (AMT). Workers were selected based on their AMT prior approval rate (>95%). Each questionnaire was presented to 3 different workers. We evaluated our proposed models on the 64 debate chat contexts. Each of the evaluated methods generated 3 samples for every chat context. To ensure calibrated ratings between systems, we show the human judges all system outputs (randomly ordered) for each particular test case at the same time. For each chat context, we conducted three kinds of assessments:

Quality Assessment Workers were provided with the following guidelines: "Given the chat context, a chat-bot needs to continue the conversation. Rate the potential answers based on your own preference on a scale of 1 to 5 (the highest):"

- 5-Excellent: "Very appropriate response, and coherent with the chat context."
- 4-Good: "Coherent with the chat context."
- 3-Fair: "Interpretable and related. It is OK for you to receive this chat response."
- 2-Poor: "Interpretable, but not related."
- 1-Bad: "Not interpretable."

In this test, the outputs of all 10 systems evaluated are then provided to worker together for a total of 30 responses. In total, we gathered $64 \cdot 30 \cdot 3 = 5760$ ratings for quality assessments, and 47 different workers participated.

Methods	Quality (MOS)	Style
vanilla-sampling	2.286 ± 0.046	—
selective-sampling	2.681 ± 0.049	10.42%
cg-ir	2.566 ± 0.048	10.24%
rank	2.477 ± 0.048	21.88%
multiply	2.627 ± 0.048	13.54%
finetune	2.597 ± 0.046	20.83%
finetune-cg-ir	2.627 ± 0.049	20.31%
finetune-cg-topic	2.667 ± 0.045	21.09%
singer-songwriter	2.373 ± 0.045	—
starwars	2.677 ± 0.048	—

Table 3: Results of quality assessments with 5-scale mean opinion scores (MOS) and JFK style assessments with binary ratings. Style results are statistically significant compared to the *selective-sampling* by paired t-tests ($p < 0.5\%$).

Style Assessment. We provided following instructions: "Which candidate responses are likely to have come from or are related to [Persona Name]?". Checkboxes were provided for the responses from style-influenced systems and from *selective-sampling* as a baseline.

Topic Assessment. The instruction was: "Which candidate answers to the chat context above are similar or related to the following answer: '[a hint topic provided by us]'?". This was also a checkbox questionnaire. Candidates are from both style- and topic-influenced systems (fine-tuned cg-topic), and from *selective-sampling* as a baseline.

Persona	Style		Topic	
	Ours	Base	Ours	Base
John F. Kennedy	21%	10%	33%	22%
Star Wars	27%	3%	14%	8%
Singer-Songwriter	31%	23%	17%	9%

Table 4: The style and topic assessments (both binary) of three models with different personas and with restriction of specific target topic for each chat context. All style and topic results are statistically significant compared to the Base (*selective-sampling*) by paired t-tests with $p < 0.5\%$.

7.4.3 Results

Overall Quality. We conducted mean opinion score (MOS) tests for overall quality assessment of generated responses with questionnaires described above. Table 3 shows the MOS results with standard error. It can be seen that all the systems based on selective sampling are significantly better than vanilla sampling baseline. When restricting output's style and/or topic, the MOS score results of most systems do not decline significantly except *singer-songwriter*, which attempts to generate lyrics-like outputs in response to to political debate questions, resulting in uninterpretable strings.

Our *rank* method uses $p(S|T)$ to pick the answer from the original persona corpus, and is thus as good at styling as the person themselves. Because most of our testing questionnaire is political, the *rank* was indeed often able to find related answers in the dataset (JFK). Also, unlike generation-based approaches, *rank* has oracle-level language fluency and it is expected to have quality score of at least 2 ("Interpretable, but not related"). Overall, however, the quality score of *rank* is still lower than other approaches. Note that a hybrid system can actually chose between rank and the decoder's outputs based on likelihood, as shown in the example of bJFk-bNixon debate in the supplemental material.

Influencing the Style. Table 3 also shows the likelihood of being labeled as JFK for different methods. It is encouraging that *finetune* based approaches have similar chances as the *rank* system which retrieves sentences directly from JFK corpus, and are significantly better than the *selective-sampling* baseline.

Influencing both Style and Topic. Table 4 summarizes the results in terms of style (the fraction of answers labeled as in-style for the target persona), and topic (the percentage of answers picked as related to the human-provided topic hint text). We used the last three of the ten listed systems, which are both styled and use specific topic hints to generate answers. These results demonstrate that it is indeed possible to provide simple prompts to a styled model and drive their answers in a desired direction while picking up the style of the persona. It also shows that the style of some characters is harder to recreate than others. For example, workers are more likely to label baseline results as lyrics from a singer-songwriter than lines from Star Wars movies, which might be because lyrics often take significant freedom with structure and grammar. We also found that it is harder for Star Wars and Singer-Songwriter bots to follow topic hints than it is for the John F. Kennedy model, largely because the political debate questions we used overlap less with the topics found in the scenting datasets for those two personas.

8 Conclusions

In this study we investigated the possibility of steering the style and content in the output of a neural encoder-decoder model[7]. We showed that acquisition of highly recognizable styles of famous personalities, characters, or professionals, is achievable, and that it is even possible to allow users to influence the topic direction of conversations. The tools described in the paper are not only useful in conversational systems (e.g., chatbots), but can also be useful as authoring tools in social media. In the latter case, the social media users might use neural models as consultants to help with crafting responses to any post the user is reading. The AMT tests show that these models do indeed provide increased recognizability of the style, without sacrificing quality or relevance.

Acknowledgments

We thank Shashank Srivastava, Donald Brinkman, Michel Galley, and Bill Dolan for useful discussions and encouragement. Di Wang is supported by the Tencent Fellowship and Yahoo! Fellowship, to which we gratefully acknowledge.

[7]The code and testing data are available at https://github.com/digo/steering-response-style-and-topic

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR* abs/1409.0.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent Dirichlet Allocation. *J. Mach. Learn. Res.* 3:993–1022.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The Mathematics of Statistical Machine Translation: Parameter Estimation. *Comput. Linguist.* 19(2):263–311.

Xing Chen, Wei Wu, Yu Wu, Jie Liu, Yalou Huang, Ming Zhou, and Wei-Ying Ma. 2017. Topic aware neural response generation. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence, February 4-9, 2017, San Francisco, California, USA.*. pages 3351–3357.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, pages 1724–1734.

Franklin C Crow. 1984. Summed-area Tables for Texture Mapping. In *Proceedings of the 11th Annual Conference on Computer Graphics and Interactive Techniques*. New York, NY, USA, pages 207–212.

Cristian Danescu-Niculescu-Mizil and Lillian Lee. 2011. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O. K. Li. 2016. Incorporating Copying Mechanism in Sequence-to-Sequence Learning .

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.* 9(8):1735–1780.

Nebojsa Jojic and Alessandro Perina. 2011. Multidimensional Counting Grids: Inferring Word Order from Disordered Bags of Words. In *Proceedings of the Twenty-Seventh Conference on Uncertainty in Artificial Intelligence*. AUAI Press, Arlington, Virginia, United States, pages 547–556.

Yuta Kikuchi, Graham Neubig, Ryohei Sasano, Hiroya Takamura, and Manabu Okumura. 2016. Controlling Output Length in Neural Encoder-Decoders. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*. pages 1328–1338.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6.

Philipp Koehn and Josh Schroeder. 2007. Experiments in Domain Adaptation for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*. Association for Computational Linguistics, Stroudsburg, PA, USA, pages 224–227.

Rik Koncel-Kedziorski, Ioannis Konstas, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2016. A theme-rewriting approach for generating algebra word problems. *CoRR* abs/1610.06210.

John D Lafferty and David M Blei. 2006. Correlated Topic Models. In Y Weiss, P B Schölkopf, and J C Platt, editors, *Advances in Neural Information Processing Systems 18*, MIT Press, pages 147–154.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A Persona-Based Neural Conversation Model. *arXiv* page 10.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and William B. Dolan. 2015. A Diversity-Promoting Objective Function for Neural Conversation Models. *Arxiv* pages 110–119.

Chia-Wei Liu, Ryan Lowe, Iulian Vlad Serban, Michael Noseworthy, Laurent Charlin, and Joelle Pineau. 2016. How NOT To Evaluate Your Dialogue System: An Empirical Study of Unsupervised Evaluation Metrics for Dialogue Response Generation. *CoRR* abs/1603.0.

Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 1412–1421.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26*. pages 3111–3119.

B O'Connor, M Krieger, and D Ahn. 2010. TweetMotif : Exploratory search and topic summarization for Twitter. *4th International AAAI Conference on Weblogs and Social Media* pages 2–3.

Alessandro Perina, Dongwoo Kim, Andrzej Turski, and Nebojsa Jojic. 2014. Skim-reading thousands of documents in one minute: Data indexing and visualization for multifarious search. In *Workshop on Interactive Data Exploration and Analytics (IDEA'14) at KDD 2014*.

Alan Ritter, Colin Cherry, and William B Dolan. 2011. Data-driven Response Generation in Social Media. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Stroudsburg, PA, USA, EMNLP '11, pages 583–593.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A Neural Attention Model for Abstractive Sentence Summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*. pages 379–389.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural Responding Machine for Short-Text Conversation. *CoRR* abs/1503.0.

Louis Shao, Stephan Gouws, Denny Britz, Anna Goldie, Brian Strope, and Ray Kurzweil. 2017. Generating Long and Diverse Responses with Neural Conversation Models .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and William B. Dolan. 2015. A Neural Network Approach to Context-Sensitive Generation of Conversational Responses. In *Naacl-2015*. Conference of the North American Chapter of the Association for Computational Linguistics Human Language Technologies (NAACL-HLT 2015), pages 196–205.

Amanda Stent, Matthew Marge, and Mohit Singhai. 2005. Evaluating Evaluation Methods for Generation in the Presence of Variation. In *Proceedings of the 6th International Conference on Computational Linguistics and Intelligent Text Processing*. Springer-Verlag, Berlin, Heidelberg, pages 341–351.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*. MIT Press, Cambridge, MA, USA, pages 3104–3112.

Orioi Vinyals and Quoc V. Le. 2015. A Neural Conversational Model. *ICML Deep Learning Workshop 2015* 37.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. In *Annual Meeting of the Association for Computational Linguistics*. pages 707–712.

Di Wang and Eric Nyberg. 2016. CMU OAQA at TREC 2016 LiveQA: An Attentional Neural Encoder-Decoder Approach for Answer Ranking. In *Proceedings of The Twenty-Fifth Text REtrieval Conference, TREC 2016, Gaithersburg, Maryland, USA, November 15-18, 2016*.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Peihao Su, David Vandyke, and Steve J. Young. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* pages 1711–1721.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR* abs/1609.0.

Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhutdinov, Richard Zemel, and Yoshua Bengio. 2015. Show, Attend and Tell: Neural Image Caption Generation with Visual Attention. *arXiv preprint arXiv:1502.03044* .