Exploring Controllable Text Generation Techniques

Shrimai Prabhumoye, Alan W Black, Ruslan Salakhutdinov

Carnegie Mellon University
Language Technologies Institute
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Who did you like the best in Avengers?
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Who did you like the best in Avengers?

Control Attributes: Robert Downey Jr. Scarlett Johansson
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Who did you like the best in Avengers?

I liked Robert Downey Jr. and Scarlett Johansson in the movie.

Control Attributes: Robert Downey Jr. Scarlett Johansson
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Did you like the movie?
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Did you like the movie?

Control Attributes: positive
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Did you like the movie?

Yeah, I loved the movie!

Control Attributes: positive
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.

Did you like the movie?

Yeah, I loved the movie!

Control Attributes: negative
Controllable Text Generation allows us to add “knobs” to control the attributes of the text to be generated.
Applications

- **Dialogue System**
  - Persona, style of responses (polite, authority), content of responses, topic of conversation
- Recommend *polite emails*
- **Story Generation**
  - plot, ending, sentiment, topic, persona
- **Report Generation** (websites, Wikipedia articles)
Motivation

Style Transfer Through Back-Translation

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, Alan W Black
Carnegie Mellon University, Pittsburgh, PA, USA
{sprabhum,ytsvetko,rsalakhu,awb}@cs.cmu.edu

Abstract
Style transfer is the task of rephrasing the text to contain specific stylistic properties without changing the impact or affect within the context. This paper introduces These goals have motivated a considerable amount of recent research efforts focused at “controlled” language generation—aiming at separating the semantic content of what is said from the stylistic dimensions of how it is said. These include approaches relying on heuristic substitu-

Published as a conference paper at ICLR 2019

Towards Controllable Story Generation

Nanyun Peng Marjan Ghazvininejad Jonathan May Kevin Knight
Information Sciences Institute & Computer Science Department University of Southern California
{npeng,ghazvinini,jonmay,knight}@isi.edu

Abstract
We present a general framework of analyzing existing story corpora to generate controllable and creative new stories. The proposed framework needs little manual annotation to achieve controllable story generation. It creates a new

Towards Content Transfer through Grounded Text Generation

Shrimai Prabhumoye
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15219
sprabhum@andrew.cmu.edu

Chris Quirk, Michel Galley
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{chrq,mgalley}@microsoft.com

Abstract
Recent work in neural generation has attracted significant interest in controlling the form of text, such as style, persona, and politeness. However, there has been less work on controlling

Plan-and-Write: Towards Better Automatic Storytelling

Lili Yao, 1, 2 Nanyun Peng, 5 Ralph Weischedel, 1 Kevin Knight, 2 Dongyan Zhao, 1 Rui Yan 1
lilyao@tencent.com, {npeng,weischedel,knight}@isi.edu
zhaodongyan.ruitian@pku.edu.cn
1 Institute of Computer Science and Technology, Peking University
2 Information Sciences Institute, University of Southern California, Tencent AI Lab

Abstract
Automatic storytelling is challenging since it requires generating long coherent natural language to describe a sensible sequence of events. Despite considerable efforts on automatic story generation in the past, prior work either is restricted in

Of WIKIPEDIA:
Knowledge-powered Conversational Agents

Emily Dinan 1, Stephen Roller 1, Kurt Shuster 1, Angela Fan, Michael Auli, Jason Weston
Facebook AI Research
{edinan,roller,kshuster,angelfan,mihaou,mjase}@fb.com

ABSTRACT

In open-domain dialogue intelligent agents should exhibit the use of knowledge, however there are few convincing demonstrations of this to date. The most popular sequence to sequence models typically “generate and hope” generic utterances

of the protagonist.

Rao and Tetreault, 2018; Xu et al., 2012; Jhamtani et al., 2017) has not focused on politeness as a style transfer task, and we argue that defining it is cumbersome. While native speakers of a language and cohabitants of a region have a good working understanding of the phenomenon of politeness

The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)
Motivation

Style Transfer Through Back-Translation

Shrimal Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, Alan W Black
Carnegie Mellon University, Pittsburgh, PA, USA
{sprabhum, ytsvetko, rsalakh, awb}@cs.cmu.edu

Abstract
Style transfer is the task of rephrasing the text to contain specific stylistic properties without changing the intent or affect within the context. This paper introduces large body of work
tools for performing style transfer. The authors describe a framework for performing style transfer, and show that their approach can be used to generate text with a wide range of stylistic properties. They also show that their approach can be used to generate text that is more natural and fluent than text generated by previous approaches.

Published as a conference paper at ICLR 2019

Towards Controllable Story Generation

Towards Content Transfer through Grounded Text Generation

Shrimal Prabhumoye
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15219
sprabhum@andrew.cmu.edu

Chris Quirk, Michel Galley
Microsoft Research
One Microsoft Way
Redmond, WA 98052
{chrisq, mgalley}@microsoft.com

Abstract
Recent work in neural generation has attracted significant interest in controlling the form of text, such as style, persona, and politeness. However, there has been less work on control-

The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)
Motivation

Large body of work

No unifying theme
Contribution

- **Controlled Generation Schema** connects prior work
  - organize prior work
  - Schema contains 5 modules
  - Identify any architecture as belonging to one of these modules
  - Schema can be used with any algorithmic paradigm

- **Collate knowledge** about different techniques
  - Insights into the advantages of techniques
  - Pave way for new architectures
  - Provide easy access to comparison
Generation Process

Encoder

h_0 \rightarrow \text{RNN} \rightarrow h_1 \rightarrow \text{RNN} \rightarrow h_2 \rightarrow \text{RNN} \rightarrow h_3 \rightarrow \text{RNN} \rightarrow h_4 \rightarrow \text{RNN} \rightarrow h_e

Are you free tomorrow?
Are you free tomorrow?
Description of the Generation Process:

1. The process begins with the Encoder, which takes an input sequence and processes it through a series of RNNs.
2. The input sequence is processed sequentially through each RNN, with the output of one RNN serving as the input to the next.
3. The output of the last RNN in the Encoder is denoted as $h_e$.
4. The $h_e$ is then fed into the Generator, which is represented by the symbol $G$.
5. The Generator processes the $h_e$ and generates an output sequence.
6. The output is compared to the input sequence to determine if the generator is correct.
7. If the output matches the input, the process concludes successfully.

Example:

- **Input:** "Are you free tomorrow?"
- **Output:** "Yes"
Are you free tomorrow?

Yes
Are you free tomorrow?

Yes

Let's
Generation Process

Encoder

$\mathbf{h}_0$ → RNN → $\mathbf{h}_1$ → RNN → $\mathbf{h}_2$ → RNN → $\mathbf{h}_3$ → RNN → $\mathbf{h}_4$ → RNN → $\mathbf{h}_e$

Decoder

$\mathbf{h}_0$ → Generator $(G)$ → $\mathbf{o}_1$ → Yes

$\mathbf{h}_1$ → Generator $(G)$ → $\mathbf{o}_2$ → let's

$\mathbf{h}_2$ → Generator $(G)$ → $\mathbf{h}_3$ → RNN → $\mathbf{h}_4$ → RNN → $\mathbf{h}_e$
Are you free tomorrow?

Yes, let's meet.

Encoder

Generator (G)

Decoder

Generator (G)

Generator (G)
Are you free tomorrow?

Yes, let's meet.
Generation Process

Encoder

RNN

$h_0$

$h_1$

$h_2$

$h_3$

$h_4$

$h_e$

Decoder

<start>

$<\text{start}>$

Generator ($G$)

Generator ($G$)

Generator ($G$)

Generator ($G$)

Yes

Yes

let's

meet

Yes

let's

meet

$<\text{end}>$

Yes

let's

meet

$<\text{end}>$
Modification Space

Decoder

Generator \((G)\)

\(<\text{start}>\)

\(x_1\)

\(h_0\)

\(o_1\)

No

Loss1

Yes

Yes

Loss1

let’s

Generator \((G)\)

\(x_2\)

\(h_1\)

\(o_2\)

cannot

No
Modification Space

External Input

Decoder

Generator ($G$)

Loss1

Yes

No

Cannot

<start>

x$_1$

h$_0$

o$_1$

x$_2$

h$_1$

o$_2$

Yes

let's

No

Yes

Loss1

Cannot
Modification Space

Decoder

Generator $(G)$

$h_0 \rightarrow x_1 \rightarrow h_1 \rightarrow o_1 \rightarrow \text{No} \rightarrow \text{Loss1}$

Generator $(G)$

$x_2 \rightarrow h_1 \rightarrow o_2 \rightarrow \text{cannot} \rightarrow \text{Loss1}$

Yes

let's
Modification Space

Sequential Input

<start>

x_1

h_0

Generator (G)

o_1

No

Loss1

Yes

x_2

h_1

Generator (G)

o_2

cannot

Loss1

Yes

let's

Decoder
Modification Space

\[
\begin{align*}
\text{Generator } (G) & \quad \text{Yes} \\
\text{Loss1} & \\
\text{Decoder} & \\
\text{Generator } (G) & \quad \text{cannot} \\
\text{Loss1} & \\
\end{align*}
\]
Modification Space

Generator

Decoder

Yes

h_1

Generator (G)

No

Loss1

Yes

x_1

h_0

x_2

h_1

Generator (G)

<start>

Yes

o_1

cannot

Loss1

let's

Generator
Modification Space

\[
\begin{align*}
\text{Generator } (G) & \quad \text{Generator } (G) \\
\text{h}_0 & \quad \text{h}_1 \\
\text{o}_1 & \quad \text{o}_2
\end{align*}
\]

<start> \quad \text{Yes} \quad \text{Yes}

\text{o}_1 \quad \text{o}_2

\text{No} \quad \text{cannot}

\text{Loss1} \quad \text{Loss1}

\text{Yes} \quad \text{let's}
Modification Space

Output

Decoder

Generator \((G)\)

\[
\begin{align*}
\text{Yes} & \quad h_0 \quad \text{Generator} \ (G) \quad h_1 \\
\text{No} & \quad \text{Loss1} \\
\text{cannot} & \quad \text{Loss1}
\end{align*}
\]
Modification Space

\[
\begin{align*}
&\text{Generator (} G \text{)} \\
&\text{Loss1}
\end{align*}
\]
Modification Space

Training Objective

Decoder

Generator \((G)\)

\(<\text{start}>\)

\(h_0\)

\(x_1\)

\(h_1\)

\(o_1\)

No

Loss1

Yes

Let’s

Loss1

Yes

\(x_2\)

Loss1

cannot

\(o_2\)

No

Yes
The proposed Schema

1. External Input \( (h_0) \) → Generator
The proposed Schema

1. External Input ($h_0$) → Generator
2. Sequential Input ($x_t$)
The proposed Schema

1. External Input ($h_0$)
2. Sequential Input ($x_t$)
3. Generator ($G$)
The proposed Schema

1. External Input ($h_0$)
2. Sequential Input ($x_t$)
3. Generator ($G$)
4. Output ($o_t$)
The proposed Schema

1. External Input ($h_0$)
2. Sequential Input ($x_t$)
3. Generator ($G$)
4. Output ($o_t$)

\[ \hat{x}_t \]
The proposed Schema

1. External Input ($h_0$)
2. Sequential Input ($x_t$)
3. Generator ($G$)
4. Output ($o_t$)

$\hat{x}_t$  
$y_t$
The proposed Schema

1. External Input ($h_0$)
2. Sequential Input ($x_t$)
3. Generator ($G$)
4. Output ($o_t$)
5. Training Objective ($\mathcal{L}$)

$\hat{x}_t$
1. **Decompose**
   - $h_e$ decomposed into subspaces
   - Provides *interpretable* representations
   - Input should contain signal of control attribute
   - Supervision on decomposed space

[Liu and Lapata (2018), Romanov et al. (2019), Balachandran et al. (2020)]
1. **Decompose**
   - $h_e$ decomposed into subspaces
   - Provides *interpretable* representations
   - Input should contain signal of control attribute
   - Supervision on decomposed space

[Liu and Lapata (2018), Romanov et al. (2019), Balachandran et al. (2020)]
1. **Decompose**

- $h_e$ decomposed into subspaces
- Provides *interpretable* representations
- Input should contain signal of control attribute
- Supervision on decomposed space

[Liu and Lapata (2018), Romanov et al. (2019), Balachandran et al. (2020)]
**External Input**

1. **Decompose**
   - $h_e$ decomposed into subspaces
   - Provides *interpretable* representations
   - Input should contain signal of control attribute
   - Supervision on decomposed space

[Liu and Lapata (2018), Romanov et al. (2019), Balachandran et al. (2020)]
2. **External Feedback**

- regularizer to control $h_e$
- must be jointly trained
- can be useful with decompose technique

[Fu et al. (2018), Wang et al. (2019a), John et al. (2019)]
2. External Feedback

- regularizer to control $h_e$
- must be jointly trained
- can be useful with decompose technique

[Fu et al. (2018), Wang et al. (2019a), John et al. (2019)]
2. **External Feedback**

- regularizer to control $h_e$
- must be jointly trained
- can be useful with decompose technique

[Fu et al. (2018), Wang et al. (2019a), John et al. (2019)]
2. External Feedback

- regularizer to control $h_e$
- must be jointly trained
- can be useful with decompose technique

3. Arithmetic or Linear Transform

4. Stochastic Changes
Sequential Input

1. Arithmetic or Linear Transform

- $\tilde{x}_t = [x_t; s]$
- $\tilde{x}_t = x_t + s$

- Changes the input to the generation itself and not the context
- Not shown promising results so far

[Noraset et al. (2017), Zhou et al. (2018), Prabhumoye et al. (2019)]
Generator Operations

1. Controlled Generator Operations
   
   - $c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t + \tanh(W_d d_t)$
   
   - $d_t = \text{dialogue act representation, change made to LSTM cell}$
   
   - Add dialogue act information in the generation process

   - $\tilde{h}_t = \tanh(W_h x_t + r_t \odot U_h h_{t-1} + s_t \odot Y g + q_t \odot (1^T Z E_{t}^{\text{new}} W))$

   - $s_t = \text{goal select gate}; q_t = \text{item select gate, GRU cell}$
   
   - recipe generation task

   [Gan et al. (2017), Kiddon et al. (2016), Wen et al. (2015)]
Generator Operations

1. Controlled Generator Operations

- \( c_t = f_t \odot c_{t-1} + i_t \odot \tilde{c}_t + \tanh(W_d d_t) \)

- \( d_t = \) dialogue act representation, change made to LSTM cell

- Add dialogue act information in the generation process

- \( \tilde{h}_t = \tanh(W_h x_t + r_t \odot U_h h_{t-1} + s_t \odot Yg + q_t \odot (1^T_L Z E_t^{new})^T) \)

- \( s_t = \) goal select gate; \( q_t = \) item select gate, GRU cell

- recipe generation task

[Gan et al. (2017), Kiddon et al. (2016), Wen et al. (2015)]
Generator Operations

2. Recurrent Neural Networks
   - LSTM, GRU

3. Transformers

4. Pre-trained language models
   - BERT, GPT-2, BART, XL-Net
1. Attention

- Focus on source sequence
- Global Attention
- Local Attention
- Multi-headed Attention

[Bahdanau et al. (2015), Luong et al. (2015), Vaswani et al. (2017)]
Generation Process

Encoder

h₀ → RNN → h₁ → RNN → h₂ → RNN → h₃ → RNN → h₄ → RNN → hₑ

<start> → Generator (G) → o₁ → Yes

Yes → Generator (G) → o₂ → Yes

Decoder
Generation Process

```
Are you free tomorrow?
```

Encoder

```
<start>
```

Generator ($G$)

```
Yes
```

Decoder

```
Yes
```

```
Are you free tomorrow?

Encoder

h₀ → RNN → h₁ → RNN → h₂ → RNN → h₃ → RNN → h₄ → RNN → hₑ

<start> → Yes → Generator (G) → a₂

h₀ → Generator (G) → o₁ → Yes

h₁ → Generator (G) → o₂

Decoder

Yes → c₂
Are you free tomorrow?

Yes

<start>
Are you free tomorrow?

Yes

<start>

let's
1. Attention

- most effective - especially self and cross
- mostly control attribute tokens have been added to source sequence for attention
- under explored for controlling attributes but has a lot of potential

[Sudhakar et al. (2019), Dinan et al. (2018), Zhang et al. (2018)]
Output

2. External Feedback
   - discriminator has to be jointly trained like GAN

3. Arithmetic or linear transform
2. External Feedback
   • discriminator has to be jointly trained like GAN

3. Arithmetic or linear transform
Training Objective

1. General Loss
   - Cross Entropy Loss
   - Unlikelihood Loss
   - Decoding Strategies
   - Used with any generation task

2. Classifier Loss
   - design multiple classifier for any control attributes

[Welleck et al. (2020), Prabhumoye et al. (2018), Yang et al. (2018)]
3. KL Divergence
   - used with stochastic changes

4. Task Specific Loss
   - design a loss for specific task (need not involve a classifier)
   - Strategy Loss
   - Coverage Loss
   - Structure Loss
Future Work

● *Empirical evaluation of schema*
  ● to understand quantitatively which modules are more effective in controlling attributes
  ● task-related architectures
  ● add these control techniques to pre-trained models like BART, T5 etc