Toolkit for Text Generation and Beyond

Zhiting Hu

09/20/2018
Outline

• Introduction to Text Generation

• Texar: A general-purpose text generation toolkit
Outline

• Introduction to Text Generation

• Texar: A general-purpose text generation toolkit
Text Generation Tasks

- Generates *natural language* from input *data or machine representations*
- Spans a broad set of natural language processing (NLP) tasks:

<table>
<thead>
<tr>
<th>Task</th>
<th>Input X</th>
<th>Output Y (Text)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chatbot / Dialog System</td>
<td>Utterance</td>
<td>Response</td>
</tr>
<tr>
<td>Machine Translation</td>
<td>English</td>
<td>Chinese</td>
</tr>
<tr>
<td>Summarization</td>
<td>Document</td>
<td>Short paragraph</td>
</tr>
<tr>
<td>Description Generation</td>
<td>Structured data</td>
<td>Description</td>
</tr>
<tr>
<td>Captioning</td>
<td>Image/video</td>
<td>Description</td>
</tr>
<tr>
<td>Speech Recognition</td>
<td>Speech</td>
<td>Transcript</td>
</tr>
</tbody>
</table>

Courtesy: Neubig, 2017
Two Central Goals

• Generating human-like, grammatical, and readable text
  • I.e., generating natural language

• Generating text that contains all relevant information inferred from inputs
  • E.g., in machine translation, the translated sentence must express the same meaning as the source sentence
Various (Deep Learning) Techniques

<table>
<thead>
<tr>
<th>Models / Algorithms</th>
<th>Other Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Neural language models</td>
<td>• Optimization</td>
</tr>
<tr>
<td>• Encoder-decoders</td>
<td>• Data pre-processing</td>
</tr>
<tr>
<td>• Seq/self-Attentions</td>
<td>• Result post-processing</td>
</tr>
<tr>
<td>• Memory networks</td>
<td>• Evaluation</td>
</tr>
<tr>
<td>• Adversarial methods</td>
<td>• ...</td>
</tr>
<tr>
<td>• Reinforcement learning</td>
<td></td>
</tr>
<tr>
<td>• Structured supervision</td>
<td></td>
</tr>
<tr>
<td>• ...</td>
<td></td>
</tr>
</tbody>
</table>
Example: Language Model

- Calculates the probability of a sentence:

  - Sentence: \( y = (y_1, y_2, ..., y_T) \)

  \[
  p_\theta(y) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{1:t-1})
  \]
Example: *Conditional* Language Model

- Conditions on additional task-dependent context $x$
  - Machine translation: (representation of) source sentence
  - Medical image report generation: (representation of) medical image

$$p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{1:t-1}, x)$$
Example: *Conditional* Language Model

- Conditions on additional task-dependent context $x$
  - Machine translation: (representation of) source sentence
  - Medical image report generation: (representation of) medical image

$$p_\theta (y \mid x) = \prod_{t=1}^{T} p_\theta (y_t \mid y_{1:t-1}, x)$$

- Language model as a decoder
Example: *Conditional* Language Model

- Conditions on additional task-dependent context $x$
  - Machine translation: (representation of) source sentence
  - Medical image report generation: (representation of) medical image

$$p_\theta(y \mid x) = \prod_{t=1}^{T} p_\theta(y_t \mid y_{1:t-1}, x)$$

- Language model as a **decoder**
- Encodes context with an **encoder**

![Diagram showing encoder and decoder with feature vector and input tokens like, this, and <BOS>]

$x$  

Encoder  

Decoder  

...
Training: Maximum Likelihood Estimation (MLE)

- Given data example \((x, y^*)\)
- Maximizes log-likelihood of the data

\[
\max_\theta \mathcal{L}_{\text{MLE}} = \log p_\theta(y^* | x)
\]

Teacher-forcing decoding:
- For every step \(t\), feeds in the ground-truth token \(y_t^*\) to decode next step
Training: Adversarial Learning

• A discriminator is trained to distinguish between real data examples and fake generated samples
• Decoder is trained to confuse the discriminator

• *Sample* \( \hat{y} \) is discrete: not differentiable
  • disables gradient backpropagation from the Discriminator to the Decoder
• Uses a differentiable approximation of \( \hat{y} \): *Gumbel-softmax decoding*
Training: Reinforcement Learning

- Optimizes test metric (e.g., BLEU) directly
- Decoder generates sample $\hat{y}$ which is used to evaluate reward
  - Greedy decoding / sample decoding / beam search decoding

![Diagram showing Policy Gradient Agent, Decoder, and BLEU metrics with rewards](image)
Various (Deep Learning) Techniques (cont’d)

- Techniques are often combined together in various ways to tackle different problems
  - An example of various model architectures

![Diagram](image)

- E refers to encoder, D to decoder, C to Classifier, A to attention, Prior to prior distribution, and M to memory
Outline

• Introduction to Text Generation

• Texar: A general-purpose text generation toolkit
Background

• Existing text generation related libraries usually focus on one particular task:
  • Machine translation: Google seq2seq (TF), OpenNMT (Pytorch), XNMT (Dynet), ...
  • Dialog: Facebook ParlAI (Pytorch)

• Other libraries/toolkits
  • For general NLP applications: AllenNLP, GluonNLP, QuickNLP, ...
  • For high conceptual-level programming: Keras, ...
  • For specific algorithms: OpenAI Gym, DeepMind Control Suite, ELF
Texar Overview

• A unified platform aiming to cover as many machine learning tasks as possible
  • Enable reuse of common components and functionalities
  • Standardize design, implementation, and experimentation
  • Encourage technique sharing among different tasks

• With an emphasis on text generation tasks
  • Provides the most comprehensive set of well-tailored and ready-to-use modules for relevant tasks

• Based on TensorFlow

• Open-source under Apache License 2.0

• Main contributors: Petuum, CMU

• Members: Zhiting Hu, Haoran Shi, Zichao Yang, Bowen Tan, Tiancheng Zhao, Junxian He, Wentao Wang, Xingjiang Yu, Lianhui Qin, Di Wang, Xuezhe Ma, Zhengzhong Liu, Xiaodan Liang, Wanrong Zhu, Devendra Singh Sachan, Eric P. Xing
Texar Highlights

Modularized
Assemble any complex model like playing building blocks

Versatile
Supports a large variety of applications/models/algorithms...

Extensible
Allows to plug in any customized or external modules
Pipeline Decomposition

• Decomposes ML models/algorithms into highly-reusable model architecture, loss, learning process, and data modules, among others

\[
\max_\theta \mathcal{L}(f_\theta, D)
\]

- learning process
- loss
- model architecture/inference process
- data
Texar Stack

Texar stack

Applications

Library APIs
Model templates + Config files

Training
Evaluation
Prediction

Models

Architectures
Losses

Library APIs
Model templates + Config files

Applications

Texar stack

Encoder
Decoder
Embedder
Classifier
(Seq) MaxLikelihood
Adversarial

MonoText
PairedText
Executor
Optimizer

Data

Dialog
Numerical
Seq/Episodic RL Agent

Losses

Rewards
RL-related
Regularize
Multi-field/type Parallel

Texar stack

lr decay / grad clip / ...
Example: Build a sequence-to-sequence model

```
# Read data
dataset = PairedTextData(data_hparams)
batch = DataIterator(dataset).get_next()

# Encode
embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
encoder = TransformerEncoder(hparams=encoder_hparams)
enc_outputs = encoder(embedder(batch['source_text_ids']),
                       batch['source_length'])

# Decode
decoder = AttentionRNNDecoder(memory=enc_outputs,
                                hparams=decoder_hparams)
outputs, length, _ = decoder(inputs=embedder(batch['target_text_ids']),
                             seq_length=batch['target_length']-1)

# Loss
loss = sequence_sparse_softmax_cross_entropy(
      labels=batch['target_text_ids'][:,1:],
      logits=outputs.logits, seq_length=length)
```

(1) Customize model template via a YAML config file

(2) Program with Texar Python Library APIs
Example: Switch between learning paradigms

(a) Maximum likelihood learning

Cross entropy loss

outputs, length, _ = decoder(  # Teacher-forcing greedy decoding
    inputs=embedder(batch['target_text_ids']),
    seq_length=batch['target_length']-1,
    decoding_strategy='train_greedy')

loss = sequence_sparse_softmax_cross_entropy(
    labels=data['target_text_ids'][[:, 1:], logits=outputs.logits, seq_length=length)

(b) Adversarial learning

(c) Reinforcement learning
Example: Switch between learning paradigms

(a) Maximum likelihood learning

Cross entropy loss

outputs, length, _ = decoder(inputs=embedder(batch['target_text_ids']),
  seq_length=batch['target_length']-1,
  decoding_strategy='train_greedy')

loss = sequence_sparse_softmax_cross_entropy(labels=data['target_text_ids'][:, 1:],
  logits=outputs.logits, seq_length=length)

(b) Adversarial learning

helper = GumbelSoftmaxTrainingHelper()  # Gumbel-softmax decoding

outputs, _, _ = decoder(helper=helper)

discriminator = Conv1DClassifier(hparams=conv_hparams)

G_loss, D_loss = binary_adversarial_losses(
  embedder(data['target_text_ids'][:, 1:]),
  embedder(soft_ids=softmax(outputs.logits)),
  discriminator)
Example: Switch between learning paradigms

(a) Maximum likelihood learning

```python
outputs, length, _ = decoder(  # Teacher-forcing greedy decoding
    inputs=embedder(batch['target_text_ids']),
    seq_length=batch['target_length']-1,
    decoding_strategy='train_greedy')

loss = sequence_sparse_softmax_cross_entropy(
    labels=data['target_text_ids'][:, 1:],
    logits=outputs.logits,
    seq_length=length)
```

(b) Adversarial learning

(c) Reinforcement learning

```python
outputs, length, _ = decoder(  # Random sample decoding
    start_tokens=[BOS]*batch_size,
    end_token=EOS,
    embedding=embedder,
    decoding_strategy='infer_sample')

agent = SeqPGAgent(
    samples=outputs.sample_id,
    logits=outputs.logits,
    seq_length=length)

for _ in range(STEPS):
    samples = agent.get_samples()
    rewards = BLEU(batch['target_text_ids'], samples)
    agent.observe(rewards)  # Train the policy (decoder)
```
Integration with any external modules

• Configuration file:
  • Insert user’s own module by specifying the python importing path

```python
rnn_cell:
  type: path.to.MyCell
  kwargs:
    my_kwarg_1: 123
    my_kwarg_2: 'xyz'
    ...
```

• Texar Python Library API
  • Fully compatible with TensorFlow-native interfaces
  • Maximum customizability
Resources

• Website: https://texar.io
• GitHub: https://github.com/asyml/texar
• Examples: https://github.com/asyml/texar/blob/master/examples
• Documentation: https://texar.readthedocs.io/
• Blog: https://medium.com/@texar