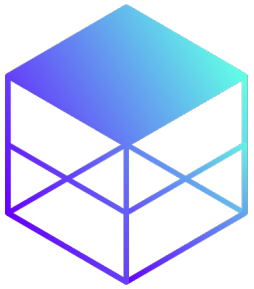


Texar

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:

Task	Input *X*	Output *Y* (Text)
Chatbot / Dialog System	Utterance	Response
Machine Translation	English	Chinese
Summarization	Document	Short paragraph
Description Generation	Structured data	Description
Captioning	Image/video	Description
Speech Recognition	Speech	Transcript

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

Texar

Various (Deep Learning) Techniques

Models / Algorithms

- Neural language models
- Encoder-decoders
- Seq/self-Attentions
- Memory networks
- Adversarial methods
- Reinforcement learning
- Structured supervision
- …

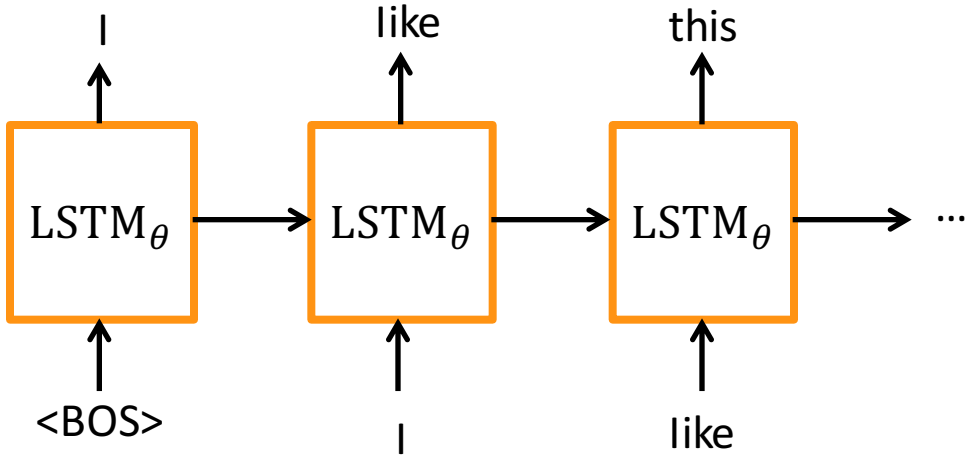
Other Techniques

- Optimization
- Data pre-processing
- Result post-processing
- Evaluation
- …

Example: Language Model

- Calculates the probability of a sentence:
  - Sentence: $\boldsymbol{y} = (y_1, y_2 \ldots, y_T)$

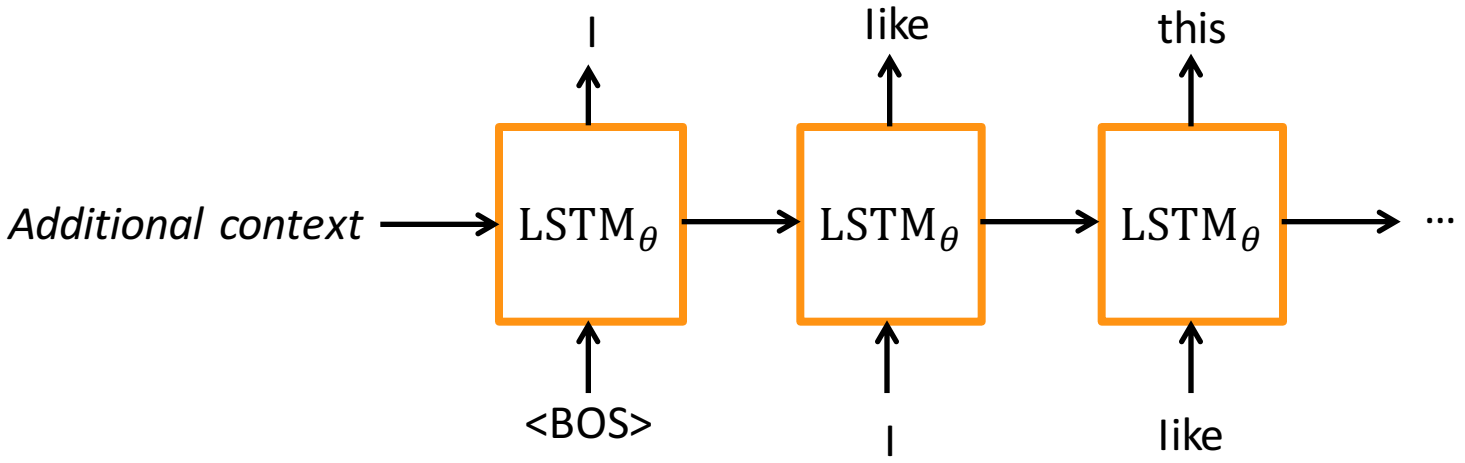
$$p_\theta(\boldsymbol{y}) = \prod_{t=1}^{T} p_\theta(y_t \mid \boldsymbol{y}_{1:t-1})$$



Example: *Conditional* Language Model

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

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



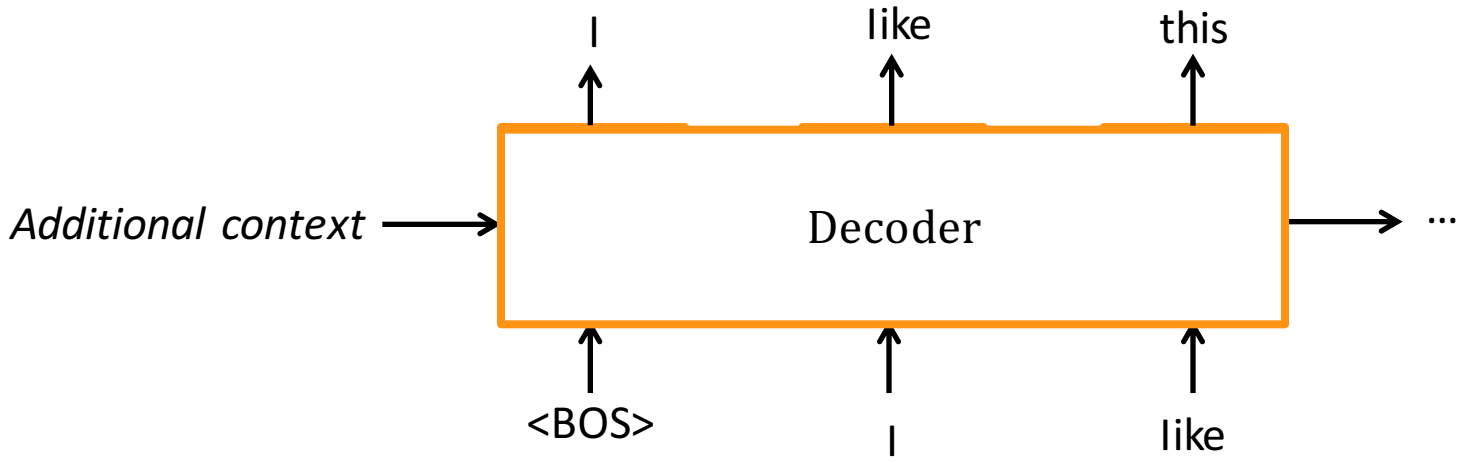
Texar

Example: *Conditional* Language Model

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

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

- Language model as a decoder



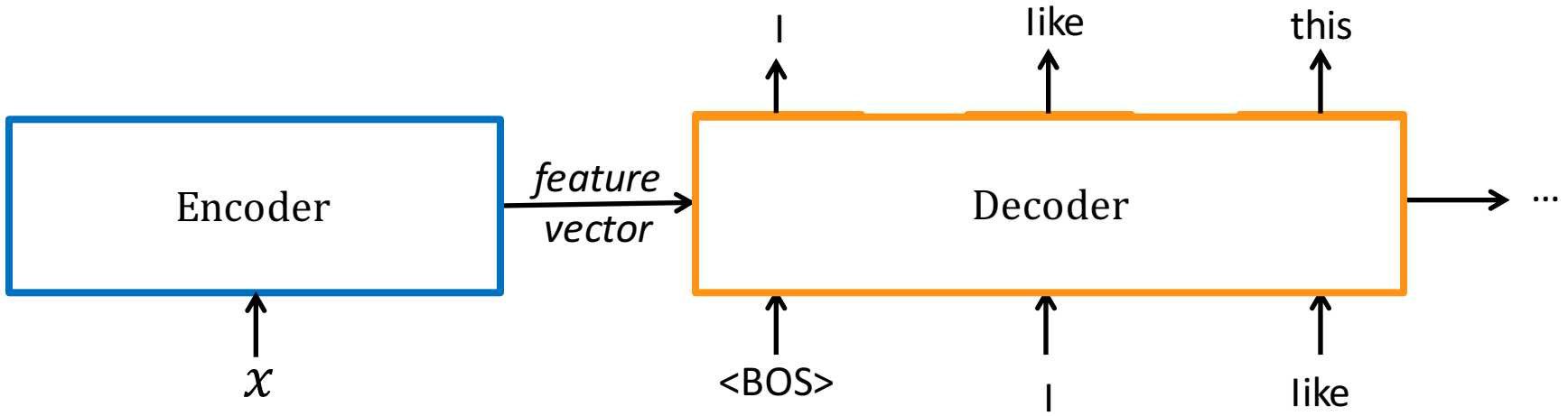
Texar

Example: *Conditional* Language Model

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

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

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



Texar

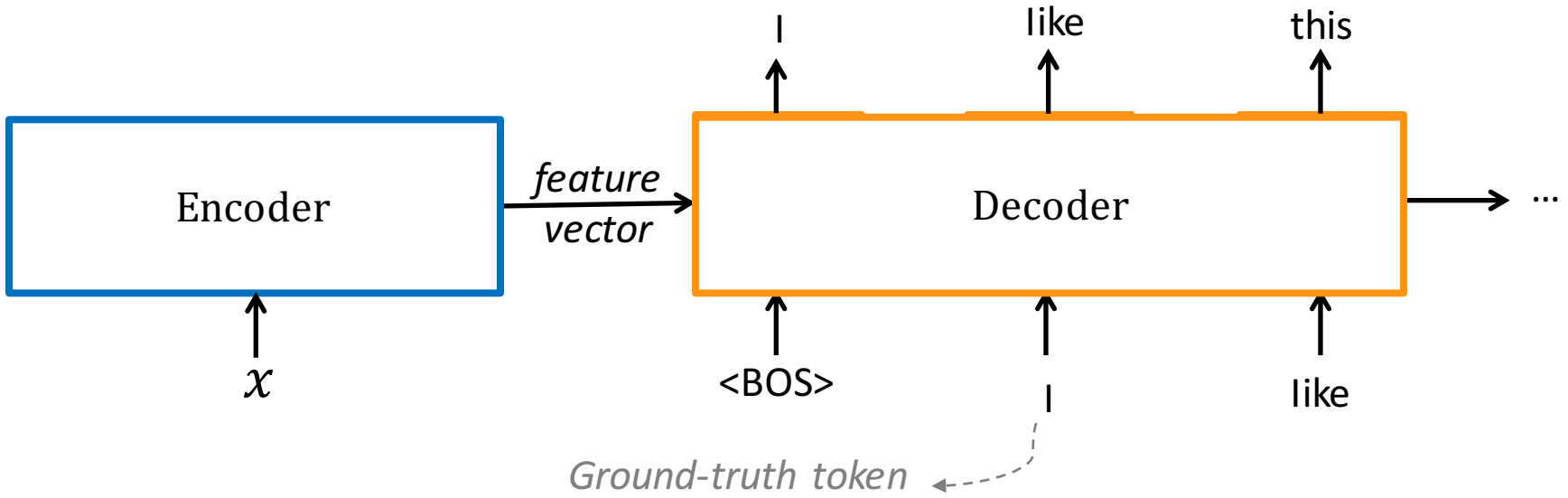
Training: Maximum Likelihood Estimation (MLE)

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

$$\max_\theta \mathcal{L}_{\text{MLE}} = \log p_\theta(\boldsymbol{y}^* \mid \boldsymbol{x})$$

Teacher-forcing decoding:

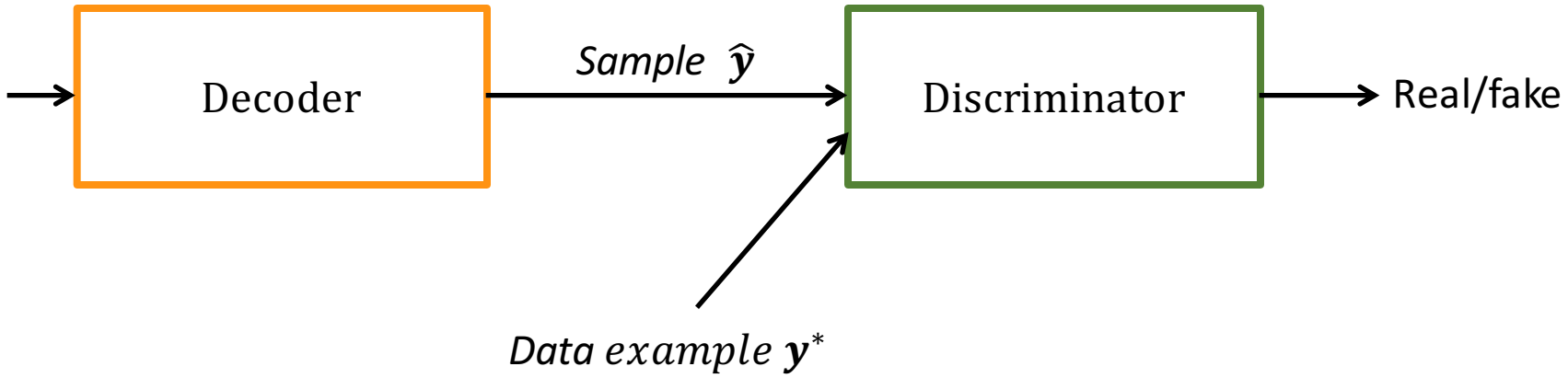
- For every step t, feeds in the ground-truth token y_t^* to decode next step



Training: Adversarial Learning

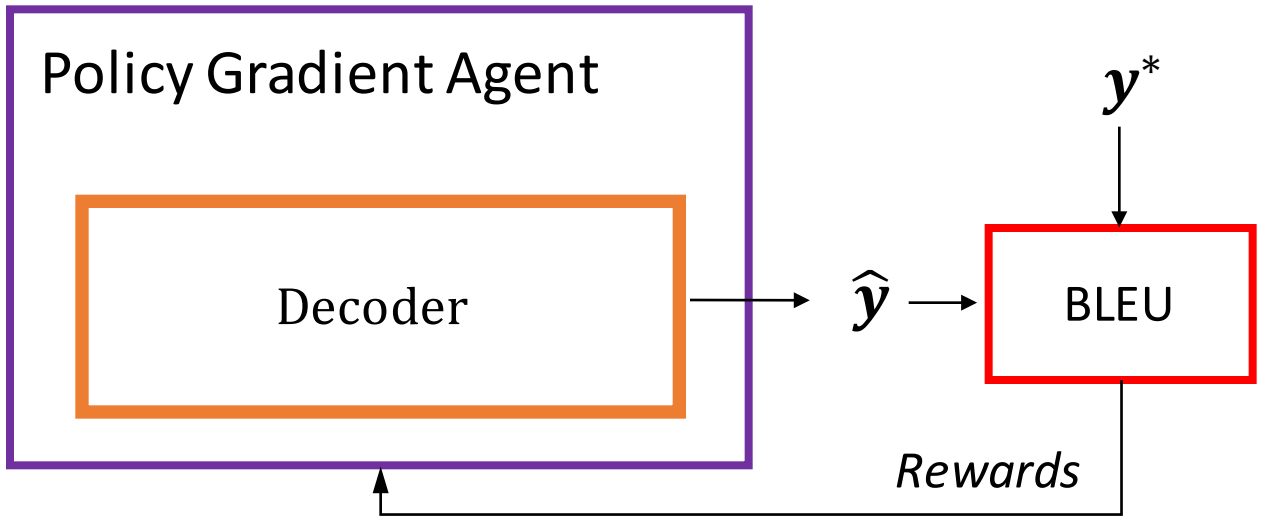
- A discriminator is trained to distinguish between real data examples and fake generated samples
- Decoder is trained to confuse the discriminator

- *Sample* $\hat{\boldsymbol{y}}$ is discrete: not differentiable
  - disables gradient backpropagation from the Discriminator to the Decoder
- Uses a differentiable approximation of $\hat{\boldsymbol{y}}$: *Gumbel-softmax decoding*



Training: Reinforcement Learning

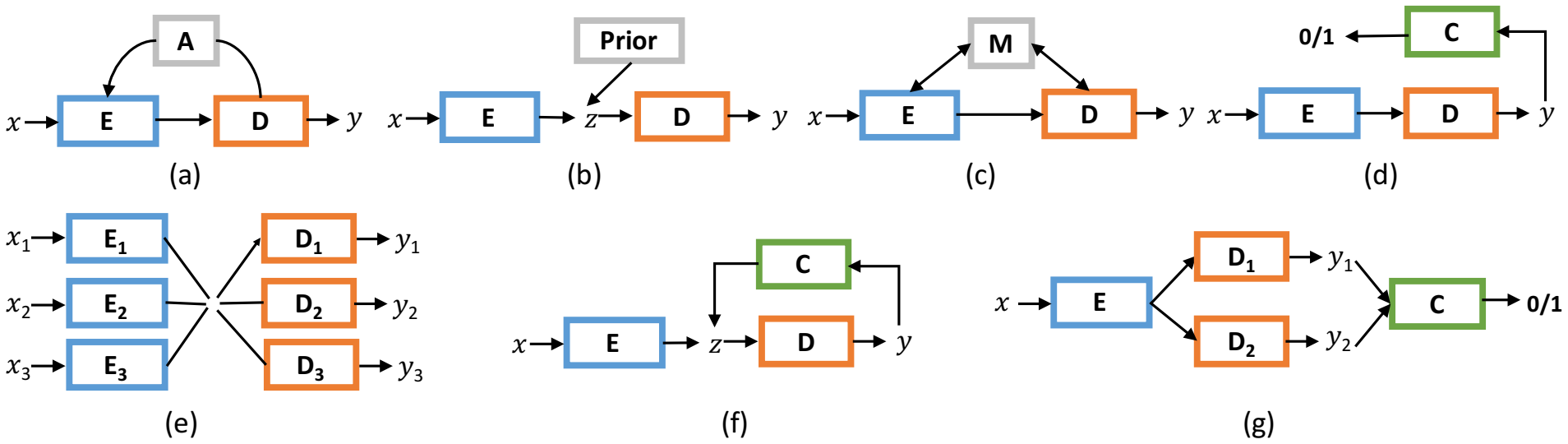
- Optimizes test metric (e.g., BLEU) directly
- Decoder generates sample $\hat{y}$ which is used to evaluate reward
  - *Greedy decoding / sample decoding / beam search decoding*



Texar

Various (Deep Learning) Techniques (cont'd)

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



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

Texar Highlights

Modularized

Assemble any complex model like playing building blocks

Versatile

Supports a large variety of applications/models/algorithms

…

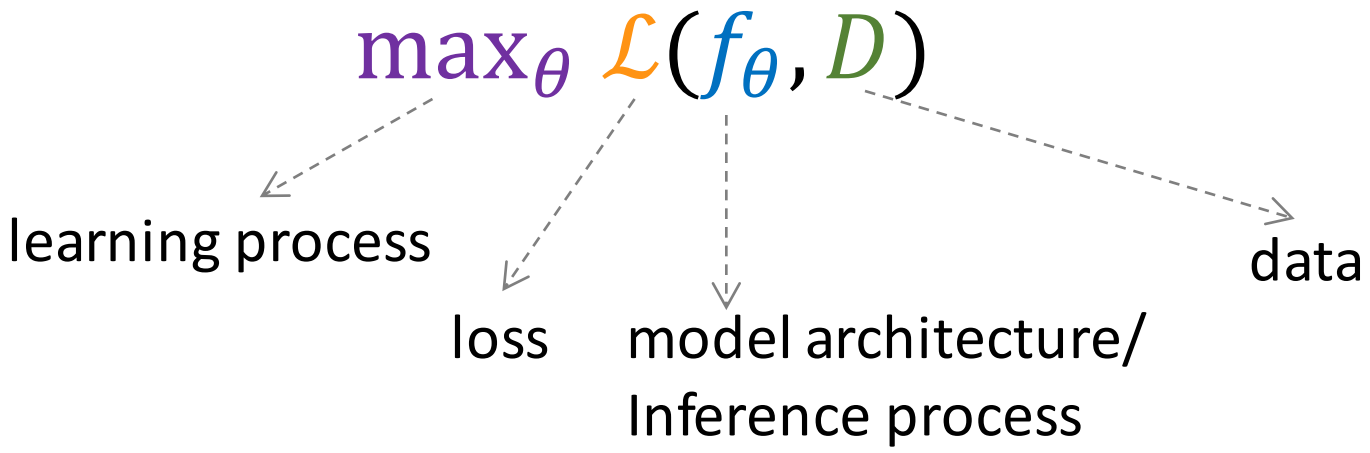
Extensible

Allows to plug in any customized or external modules

Texar

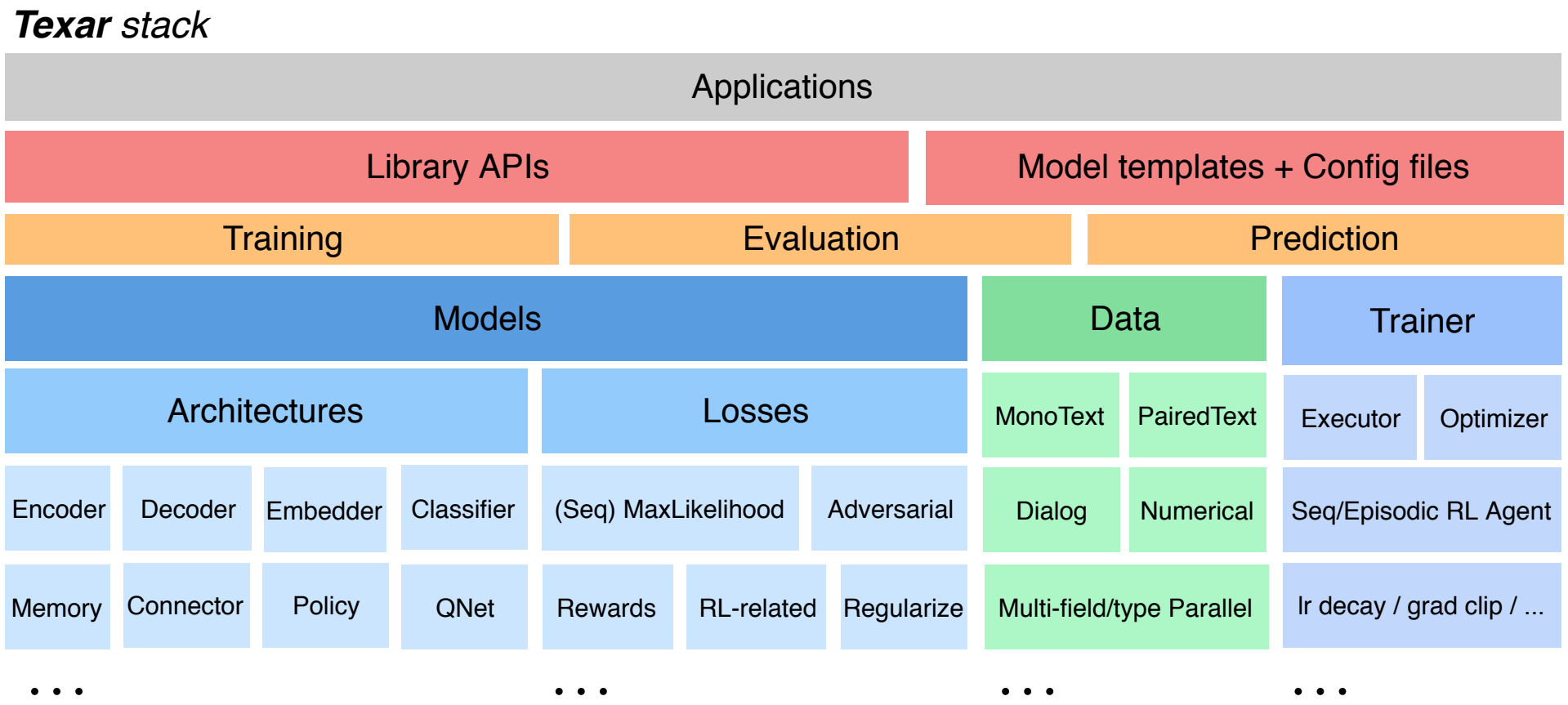
Pipeline Decomposition

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



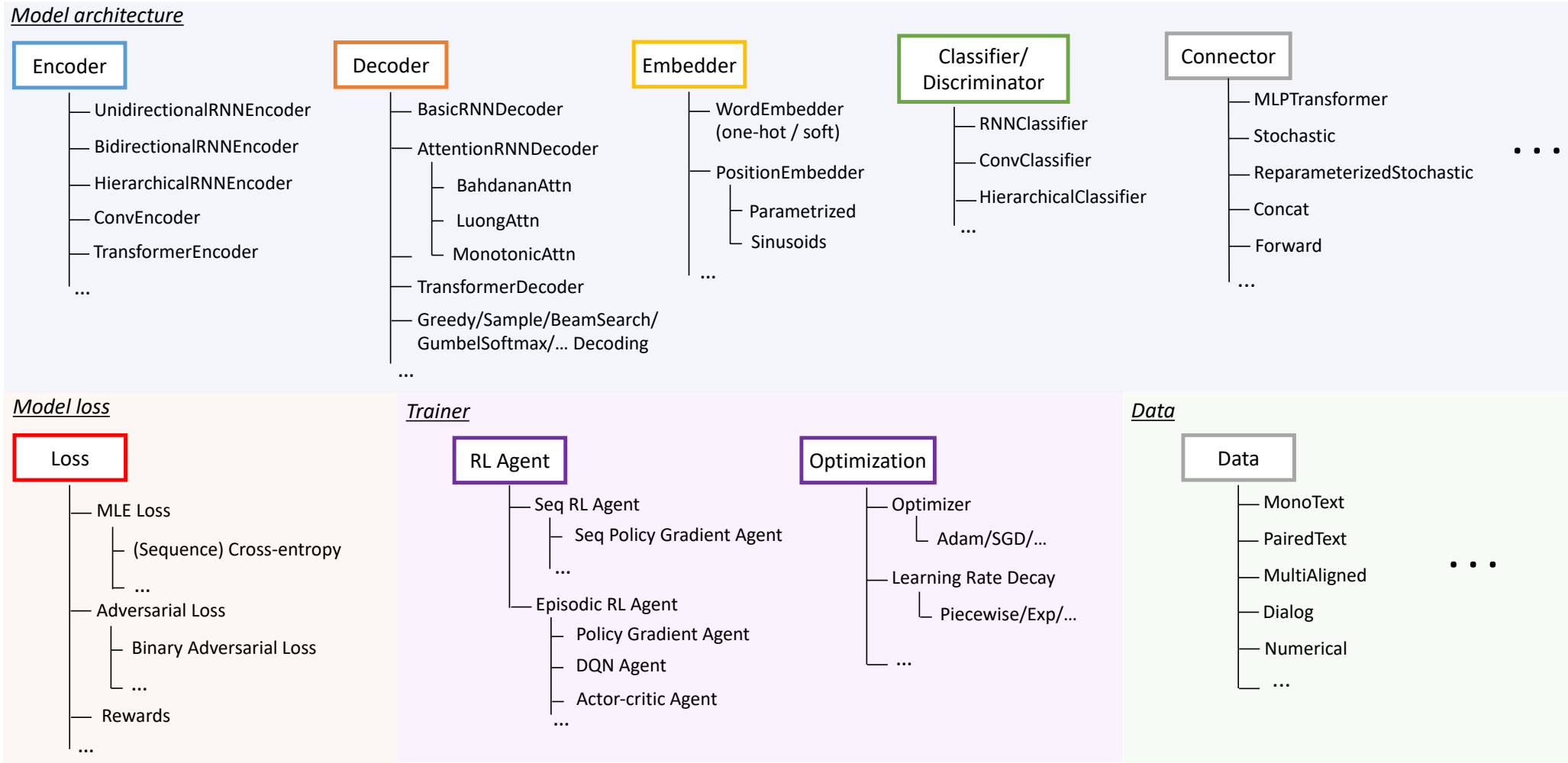
Texar

Texar Stack



Texar

Module Catalog



Texar

Example: Build a sequence-to-sequence model

```yaml
source_embedder: WordEmbedder
source_embedder_hparams:
  dim: 300
encoder: UnidirectionalRNNEncoder
encoder_hparams:
  rnn_cell:
    type: BasicLSTMCell
    kwargs:
      num_units: 300
    num_layers: 1
    dropout:
      output_dropout: 0.5
      variational_recurrent: True
embedder_share: True
decoder: AttentionRNNDecoder
decoder_hparams:
  attention:
    type: LuongAttention
beam_search_width: 5
optimization: ...
```

(1) Customize model template via a YAML config file

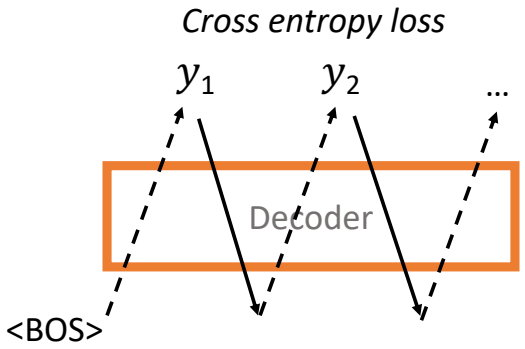
```python
# Read data
dataset = PairedTextData(data_hparams)
batch = DataIterator(dataset).get_next()

# 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)

```

(2) Program with Texar Python Library APIs

Texar

Example: Switch between learning paradigms

(a) Maximum likelihood learning

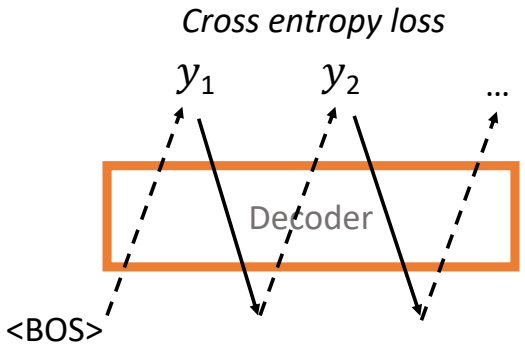



```
outputs, length, _ = decoder(          # Teacher-forcing greedy decoding
    inputs=embedder(batch['target_text_ids']),
    seq_length=batch['target_length']-1,
    decoding_strategy='train_greedy')

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

Texar

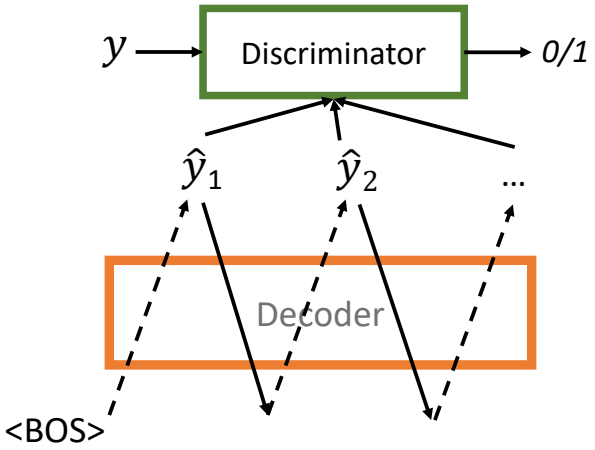
Example: Switch between learning paradigms

(a) Maximum likelihood learning




```
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

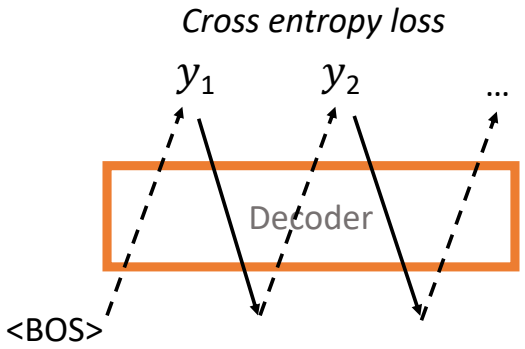



```
helper = GumbelSoftmaxTrainingHelper(          # Gumbel-softmax decoding
    start_tokens=[BOS]*batch_size, end_token=EOS, embedding=embedder)
outputs, _, _ = decoder(helper=helper)

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)
```

Texar

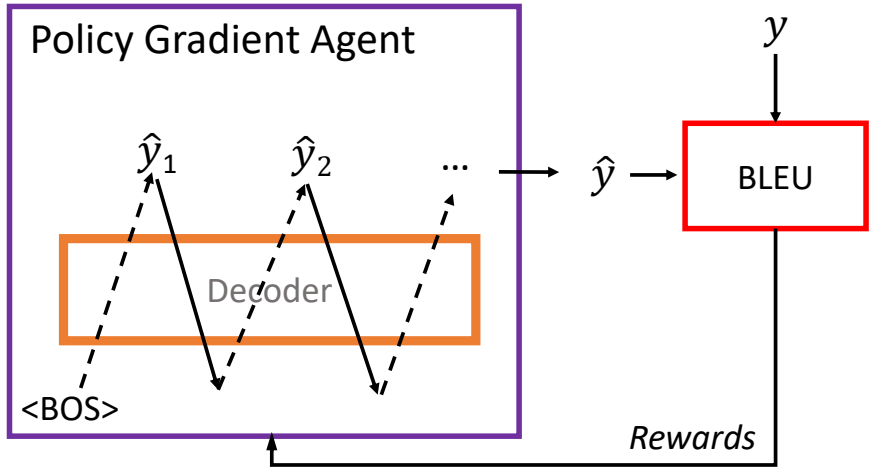
Example: Switch between learning paradigms

(a) Maximum likelihood learning




```
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)
```

(c) Reinforcement learning




```
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

```
7  rnn_cell:
8    type: path.to.MyCell
9    kwargs:
10     my_kwarg_1: 123
11     my_kwarg_2: 'xyz'
12     …
```

- 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
- Tech report: https://arxiv.org/pdf/1809.00794.pdf