UC San Diego

MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

Carnegie Mellon University School of Computer Science

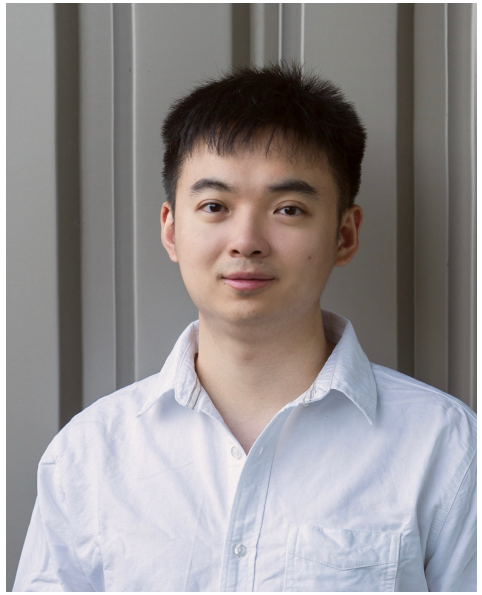
Petuum

A "Standard Model" for Machine Learning

Zhiting Hu, Hao Zhang, Eric Xing

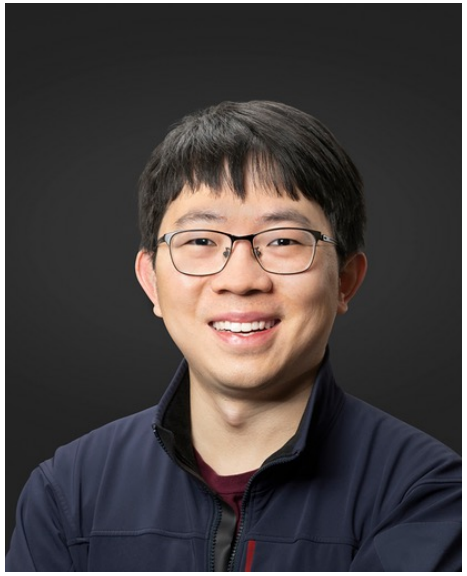
DeepLearn 2023 Winter

Presenters



Zhiting Hu

Assistant Professor
@UCSD



Hao Zhang

Assistant Professor
@UCSD

Eric P. Xing

Professor @ CMU
President @ MBZUAI
Co-founder @ Petuum

UC San Diego

MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

Carnegie Mellon University School of Computer Science

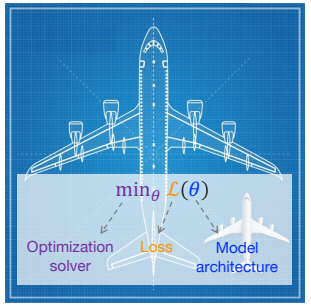
Petuum

Schedule

- **Lecture#1:** Theory: The Standard Model of ML

  A blueprint of ML paradigms for ALL experience

  (Jan 19 Thursday, 4:45pm-6:15pm UK Time)



- **Lecture#2:** Tooling: Operationalizing The Standard Model

  Compose your ML solutions like playing Lego

  (Jan 20 Thursday, 1:00pm-2:30pm)

- **Lecture#3:** Computing: Modern infrastructure for productive ML

  Automatic tuning, distributing, and scheduling

  (Jan 20 Thursday, 4:45pm-6:15pm)

Tooling:

Operationalizing The “Standard Model”

$$\min_{q,\theta} -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

Recap of Part-I: Standard Equation

$$\min_{q,\theta} -\mathbb{E}_{q(\boldsymbol{x},\boldsymbol{y})}\Big[f(\boldsymbol{x},\boldsymbol{y})\Big] + \beta\mathbb{D}\Big(q(\boldsymbol{x},\boldsymbol{y}), p_\theta(\boldsymbol{x},\boldsymbol{y})\Big) - \alpha\mathbb{H}(q)$$

3 terms:

Experience
- *data examples*
- *rules*
- *reward*
- *adversarial models*
- …

Divergence
- *Cross Entropy*
- *JS Divergence*
- *f-Divergence*
- *Wasserstein Dist.*
- …

Uncertainty
e.g., Shannon entropy

$$\min_{q,\theta} \quad -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

Recap of Part-I: Standard Equation

$$\min_{q,\theta} \quad -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{E})$$

- Optimization solver
- Loss
- Model architecture
- Experience

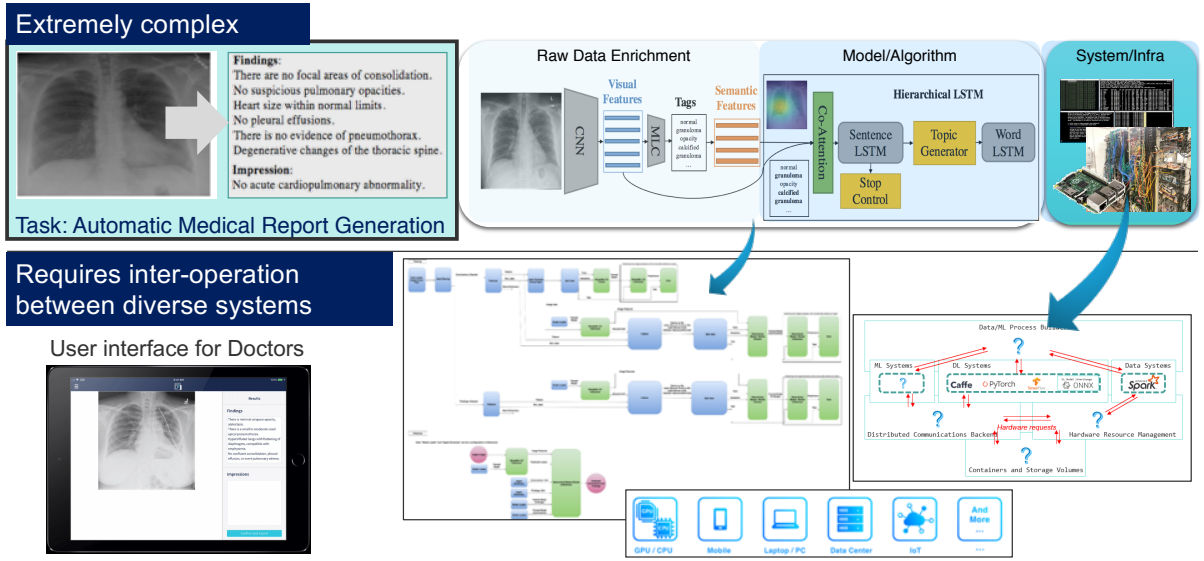
Part-II: Operationalizing The "Standard Model"

$$\min_{q,\theta} \; -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

$$\min_{\theta} \mathcal{L}(\theta, \mathcal{E})$$

Optimization solver | Loss | Model architecture | Experience

- ML solution design
  - Plugging arbitrary experiences in learning
- Tooling for composable ML



Problem solving by plugging arbitrary available experiences in learning

Standard equation

$$\min_{q,\theta} -\mathbb{E}_{q(\boldsymbol{x},\boldsymbol{y})}\Big[f(\boldsymbol{x},\boldsymbol{y})\Big] + \alpha\mathbb{D}\Big(q(\boldsymbol{x},\boldsymbol{y}), p_\theta(\boldsymbol{x},\boldsymbol{y})\Big) - \beta\mathbb{H}(q)$$

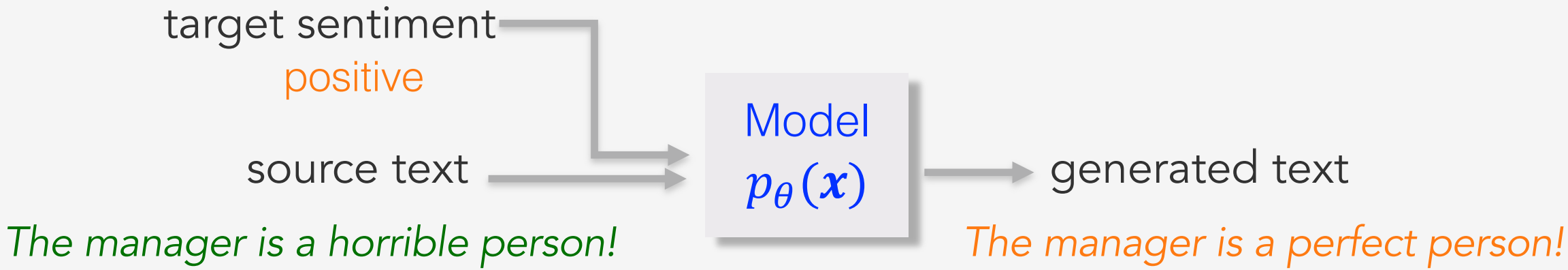
$$f = w_1 \cdot f(\boldsymbol{x}\,|\,\text{🗄}) + w_2 \cdot f(\boldsymbol{x}\,|\,\text{📖}) + w_3 \cdot f(\boldsymbol{x}\,|\,\text{💲}) + w_4 \cdot f(\boldsymbol{x}\,|\,\text{👍}) + \cdots$$

Problem: Controllable Text Generation

Ex. controlling sentiment

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

Key challenge: no direct supervision data

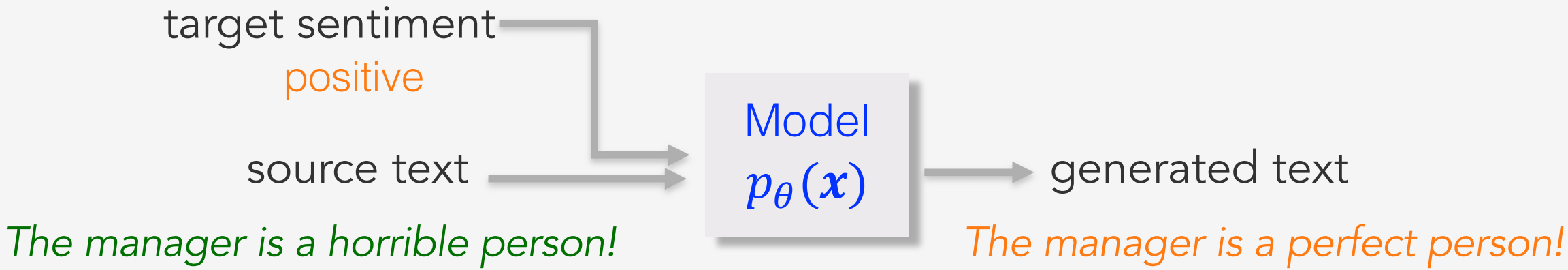


[Hu et al., ICML'17]

Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use



[Hu et al., ICML'17]

Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use

Sentiment classifier

The manager is a perfect person! →  *Sentiment classifier* → *logit of target sentiment*

[Hu et al., ICML'17]

Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use

Sentiment classifier

[Hu et al., ICML'17]

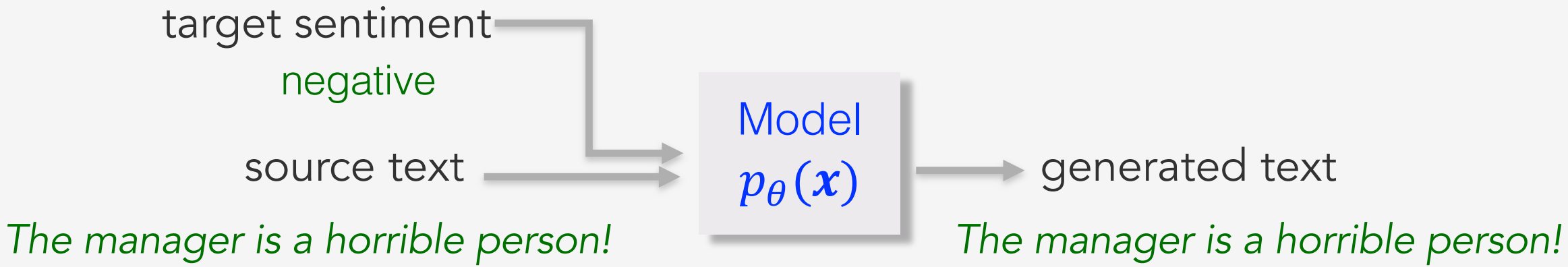
Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use

Sentiment classifier

Self-construction data examples



[Hu et al., ICML'17]

Designing the Solution

Goals: generating a new sentence that has the target sentiment while preserving all other aspects

1) Consider what experiences to use

Sentiment classifier

Self-construction data examples

Language model

…

The manager is a perfect person! →  *Language model* → *perplexity score*

[Hu et al., ICML'17]

Designing the Solution

1) Consider what experiences to use

Sentiment classifier

Self-construction data examples

Language model …

2) Plug experiences into the algorithm

[Hu et al., ICML'17]

Designing the Solution

1) Consider what experiences to use

Sentiment classifier

Self-construction data examples

Language model …

2) Plug experiences into the algorithm

Source text: The manager is a horrible person!

Experiences plugged in:

$$f = f(\boldsymbol{x}|\,\text{S}\,)$$

$$+\ f(\boldsymbol{x}|\,\text{DB}\,)$$

$$+\ f(\boldsymbol{x}|\,\text{L}\,)$$

Resulting generated text:

good good good good …

The manager is a delicious person!

The manager is a perfect person!

[Hu et al., ICML'17]

Experimental Results

	Accuracy (↑)	Perservation (↑)	Language quality (↓)
Sentiment classifier 	**98.9** 😁	0.1	326.1
+ Self-construction data examples 	87.7 😁	**65.6** 😁	115.6
+ Language model 	91.2 😁	57.8 😁	**47.2** 😁

[Hu et al., ICML'17]

Applications in other controllable generation problems

Controlling sentiment

Pos: The film is full of imagination!

Neg: The film is strictly routine!

[Hu et al., ICML'17]

Rewriting content

He scored 23 points and pulled down 8 rebounds .

Name	LeBron James
Points	32
Rebounds	4
Assists	7

LeBron James scored 32 points, pulled down 4 rebounds, and added 7 assists.

[Lin et al., pre-print'20]

Guiding conversation flow

Hi, how are you today?

Fine. Just finished riding along the river.

Cool! You can ride bikes, listen to music there too.

Yes. I like Taylor Swift.

I love to sing her songs! Do you?

Not really. I cannot sing well.

How about dancing? I love dancing!

[Tang et al., ACL'19]

Applications in other controllable generation problems

Source

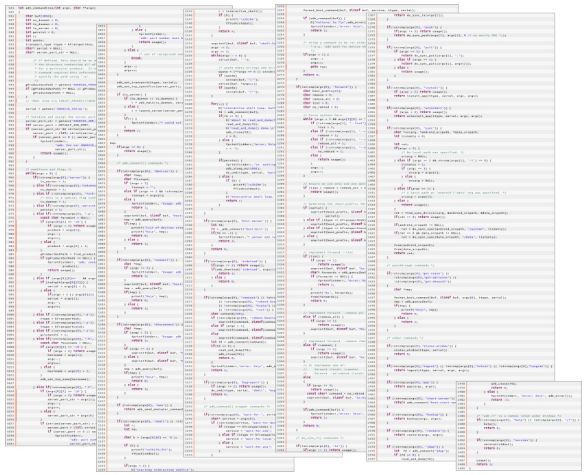
Generated images under different poses

Constraints of human gesture

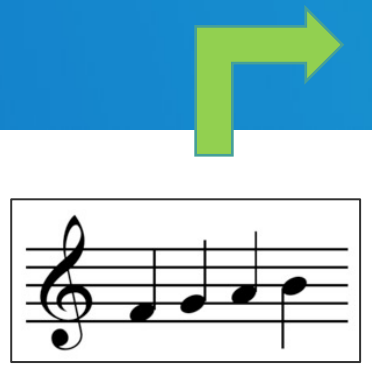
source *target pose* *Base* *+ Constraint* *true target*

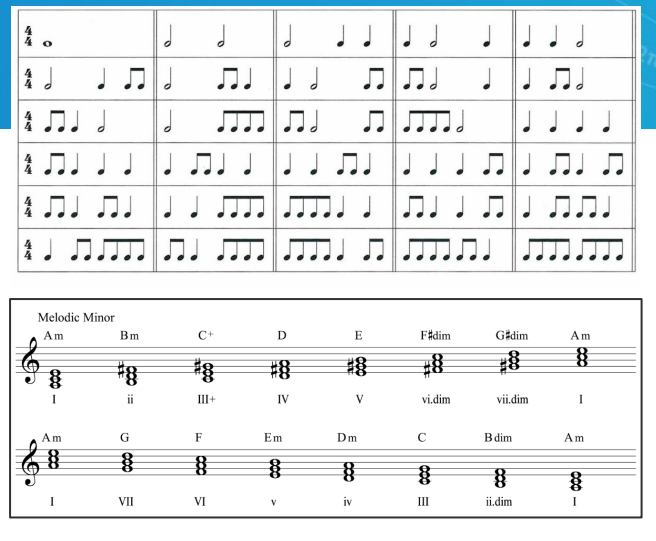
[Hu et al., NeurIPS'18]

Compositionality



One-off design and programming





Modular and standardized

Complex “rhythms” and “chords” systematically built out of simpler “notes”

Use MT as the running example due to time constraint

Running Example: Machine Translation (MT)

cleaning, tokenizing, truncating, …

source.dat

I like this movie.

Lovely and poignant

Insanely hilarious!

…

target.dat

Ich mag diesen film.

Schön und ergreifend

Wahnsinnig witzig!

…

data instances

BLEU

Semantic ROUGE

score

reward

Adversaries

experiences

model architecture

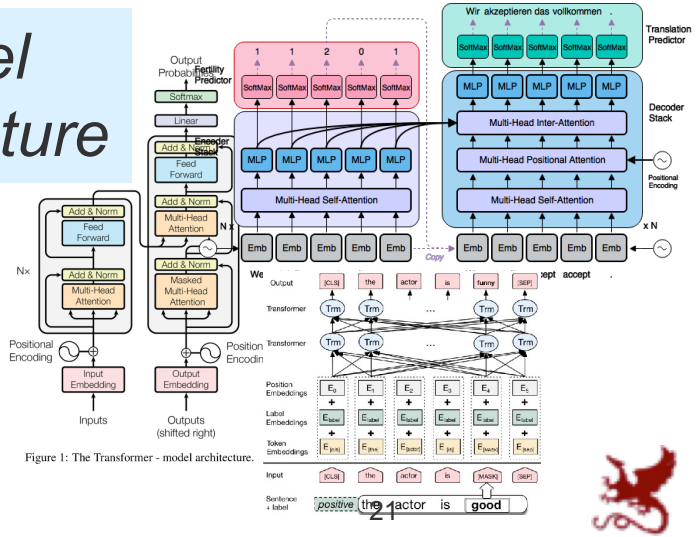


Figure 1: The Transformer - model architecture.

training

$$\min_{q,\theta} \; -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

evaluation post-processing

…

UC San Diego

MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

Running Example: Machine Translation (MT)

source.dat

target.dat

cleanin

truncat

BLEU

Semantic

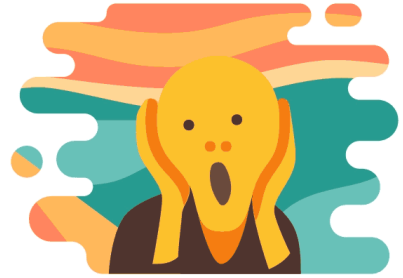
score

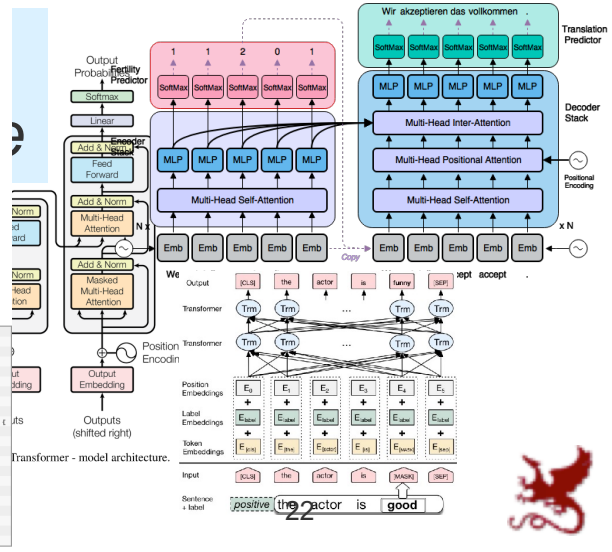
ROU

evaluation

post-processing

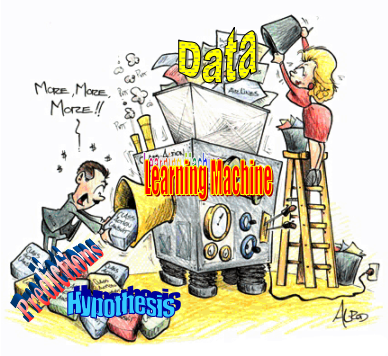
...

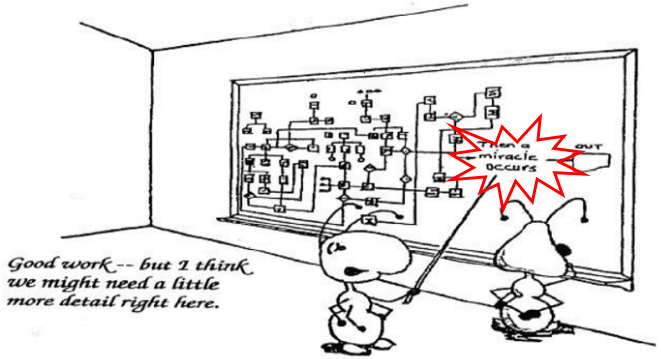


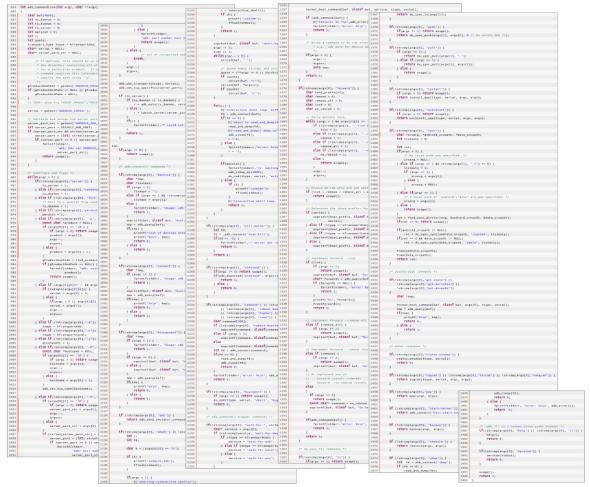


Petuum

Solution with Composable ML Tools







Composable ML

A **modularized** way to build complex applications

Solution with Composable ML Tools

- Build AI solutions more easily, via pick-and-choose:
  - Experiences: data instances, reward, adversaries, rules, …
  - Models: recurrent, transformer, convolutional, …
  - Tasks: classification, regression, generation, discovery, …
- Stop writing same one-off code again and again
  - More reliable and easier to debug
  - Easier to onboard new developers

$\min_\theta \mathcal{L}(\theta, \mathcal{E})$

Running Example: Machine Translation (MT)

cleaning, tokenizing, truncating, …

source.dat

I like this movie.

Lovely and poignant

Insanely hilarious!

…

target.dat

Ich mag diesen film.

Schön und ergreifend

Wahnsinnig witzig!

…

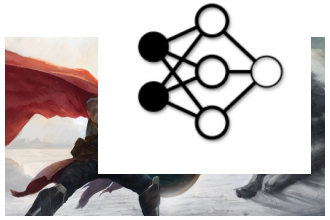
data instances

BLEU

Semantic ROUGE

score

reward



Adversaries

experiences

model architecture

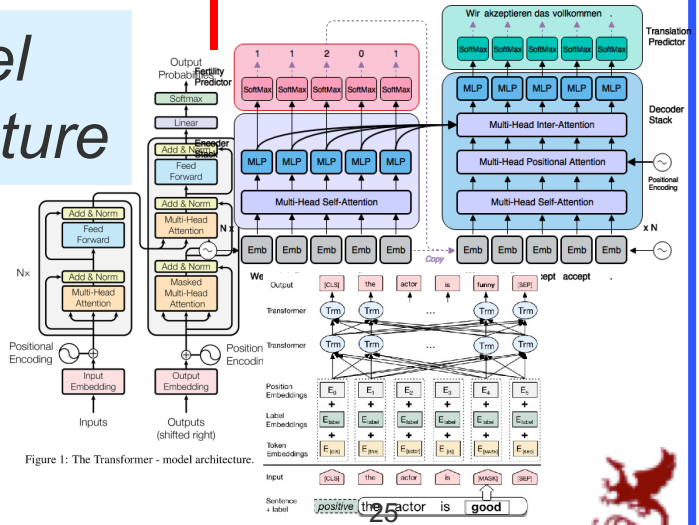


Figure 1: The Transformer - model architecture.

training

$$\min_{q,\theta} \; -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

evaluation post-processing

…

ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture

ML Components

Optimization

Loss

Operation

Architecture

Divergence

Experience

Architecture

- Neural network design
- Graphical model design
- Compositional architectures

Architecture

- Neural network design
- Graphical model design
- Compositional architectures

Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
  - Sentence:

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$$

Example:

(I, like, this, …)

Neural Architecture (1): 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:

(I, like, this, …)

$$\cdots p_\theta\,(like \mid I)\; p_\theta(this \mid I, like) \cdots$$

Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
  - Sentence:

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$$

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

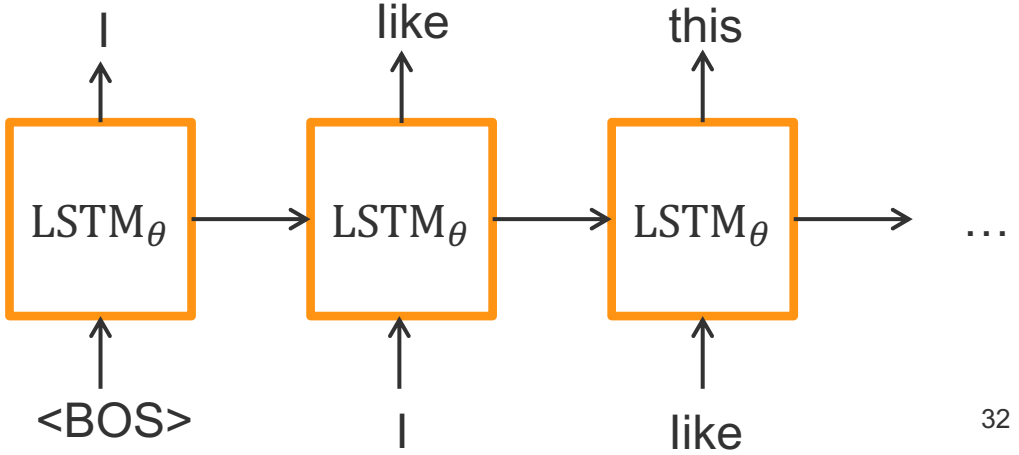
Example:

(I, like, this, …)

$$\cdots p_\theta\,(like \mid I)\; p_\theta(this \mid I, like) \cdots$$

Architecture (1.1)

LSTM RNN



Neural Architecture (1): Language Model

- Calculates the probability of a sentence:
  - Sentence:

$$\boldsymbol{y} = (y_1, y_2, \ldots, y_T)$$

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

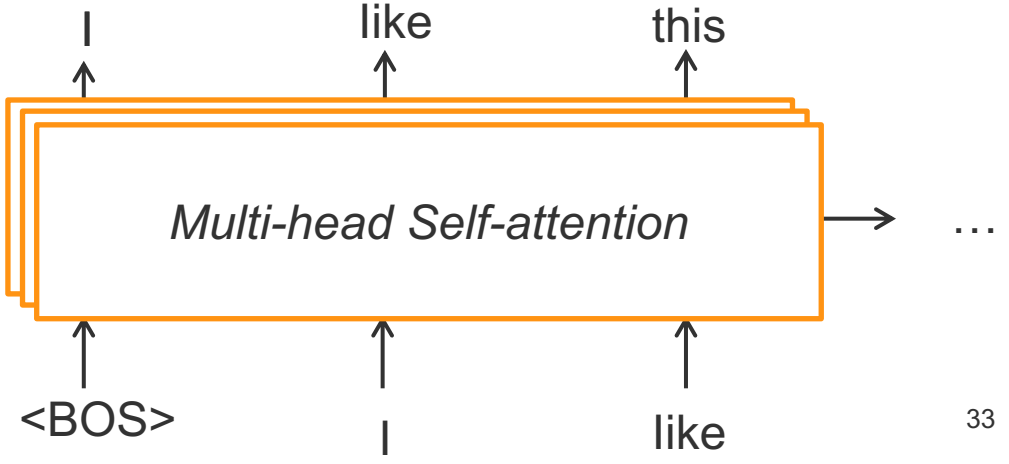
Example:

(I, like, this, …)

$$\cdots p_\theta\,(like \mid I)\; p_\theta(this \mid I, like) \cdots$$

Architecture (1.2)

Transformer

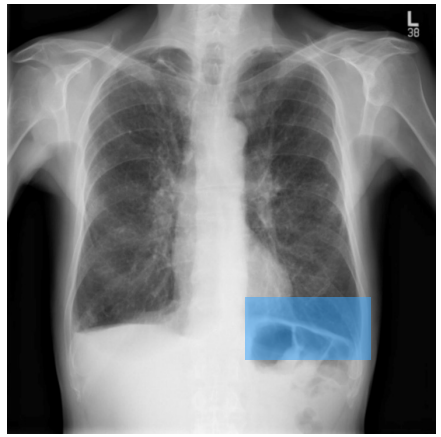


Neural Architecture (2): Conditional Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$
  - Machine translation: source sentence

I like this movie. ⟶ Ich mag diesen film.

  - Medical image report generation: medical image



⟶ … There is chronic pleural-parenchymal scarring within the lung bases. No lobar consolidation is seen. …

Neural Architecture (2): Conditional Language Model

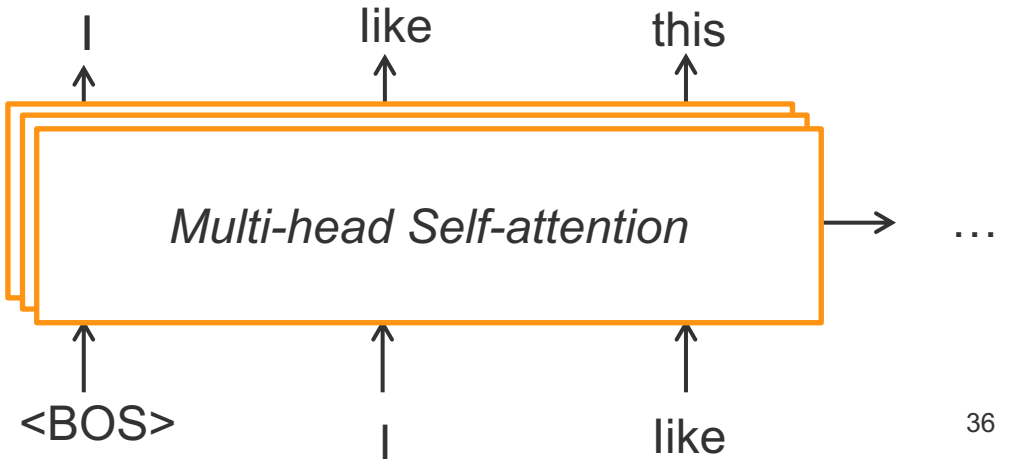
- Conditions on additional task-dependent context $\boldsymbol{x}$

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

Neural Architecture (2): Conditional Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$

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

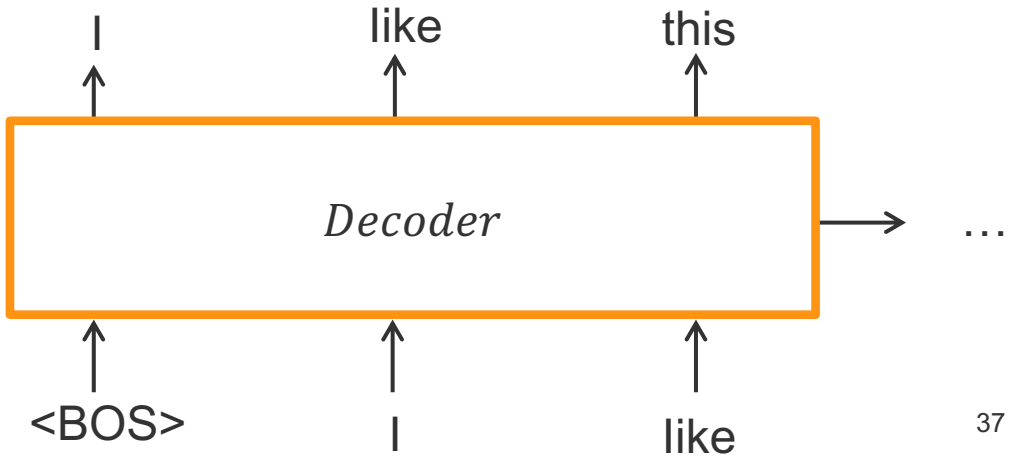


Neural Architecture (2): Conditional Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$

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

- Language model as a decoder

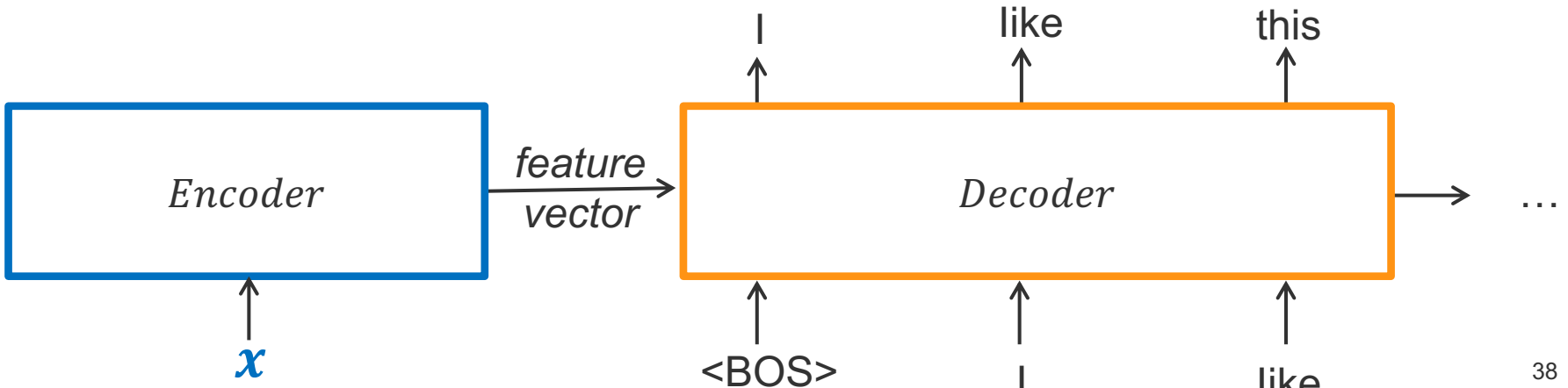


Neural Architecture (2): Conditional Language Model

- Conditions on additional task-dependent context $\boldsymbol{x}$

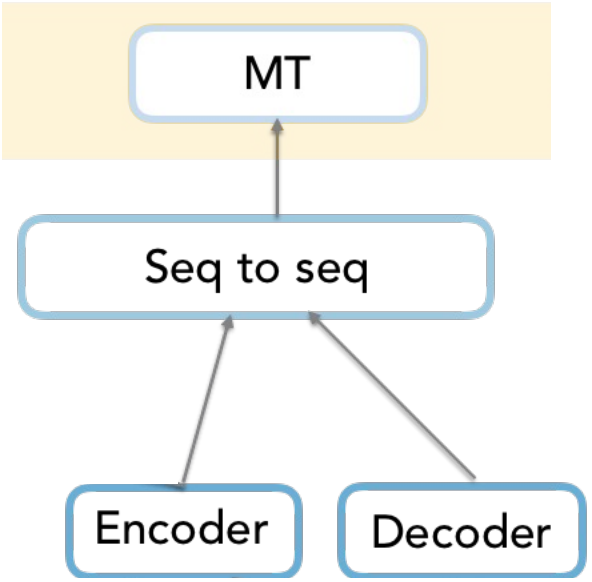
$$p_\theta(\boldsymbol{y} \mid \boldsymbol{x}) = \prod_{t=1}^{T} p_\theta(y_t \mid \boldsymbol{y}_{1:t-1}, \boldsymbol{x})$$

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



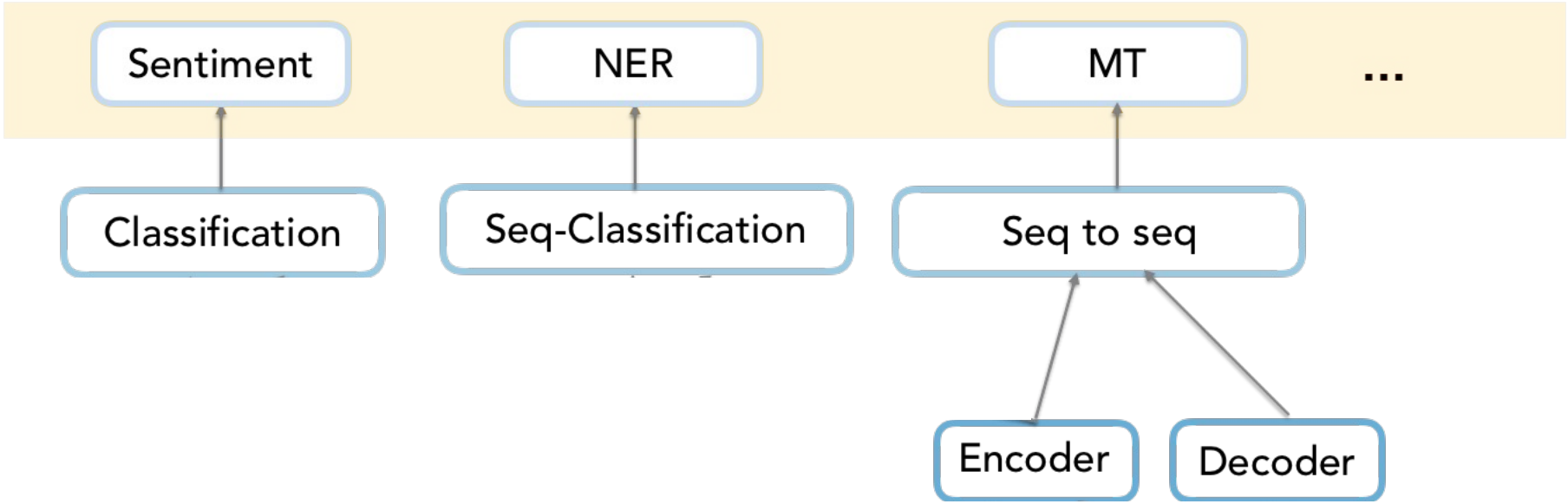
Neural Architecture Components

Task:



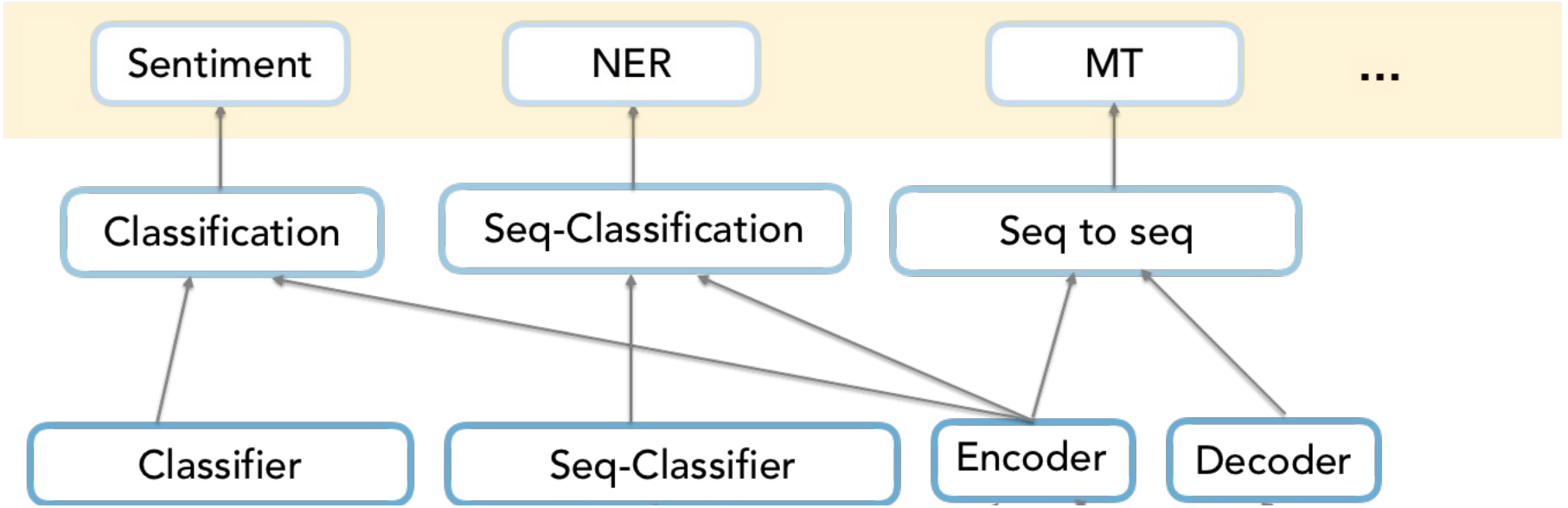
Neural Architecture Components

Task:



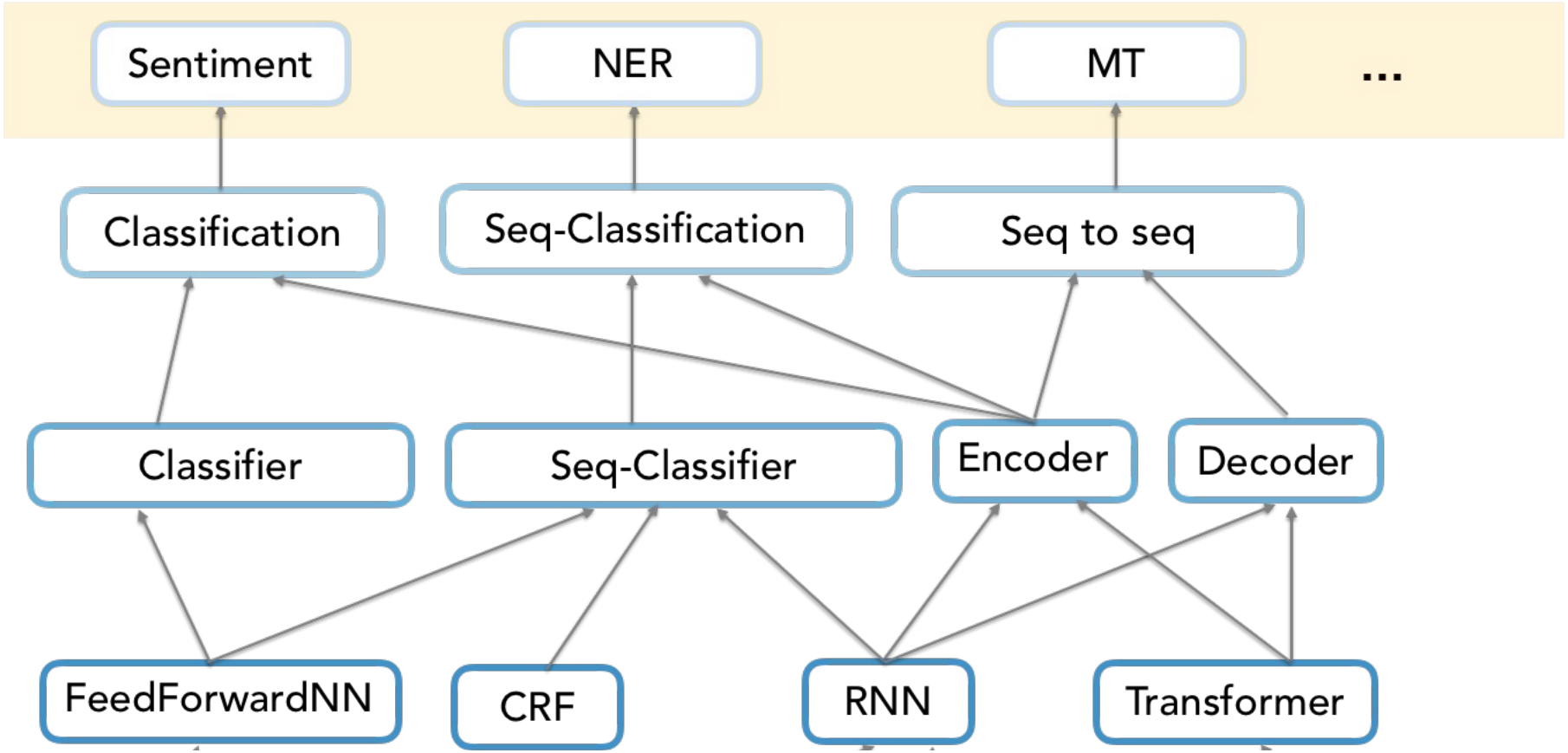
Neural Architecture Components

Task:



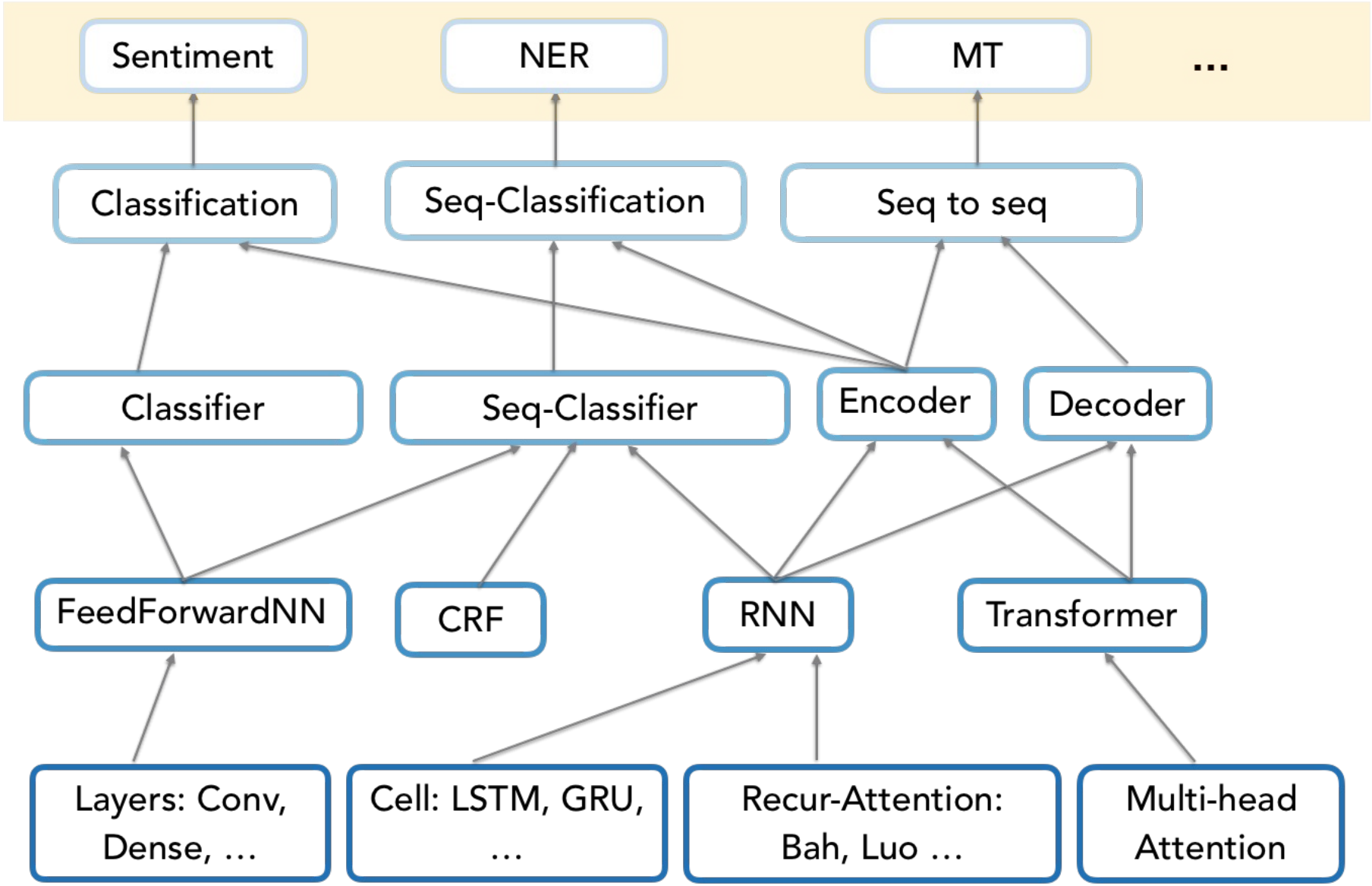
Neural Architecture Components

Task:



Neural Architecture Components

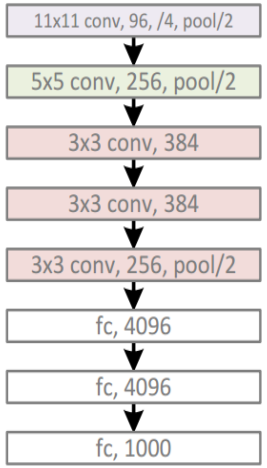
Task:



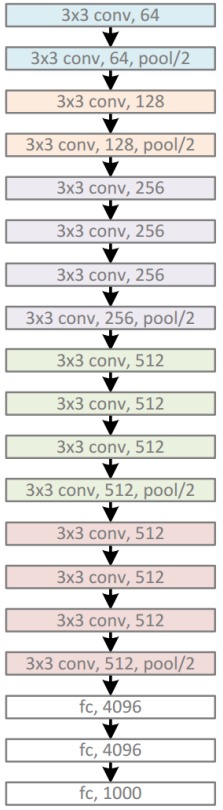
Other Neural Architecture Components

- Layers
  - Fully connected
  - Convolutional & pooling
  - Recurrent
  - ResNets
  - Etc.
- Activation functions
  - Linear and ReLU
  - Sigmoid and tanh
  - Etc.

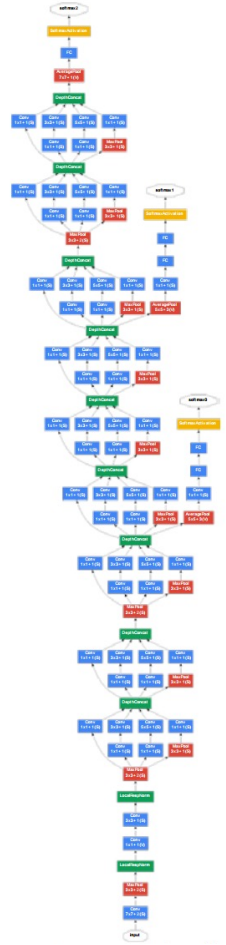
AlexNet
8 layers



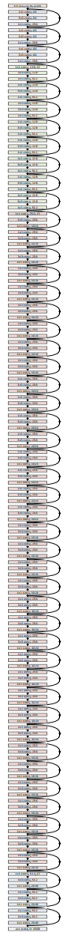
VGG
19 layers



GoogleNet
22 layers

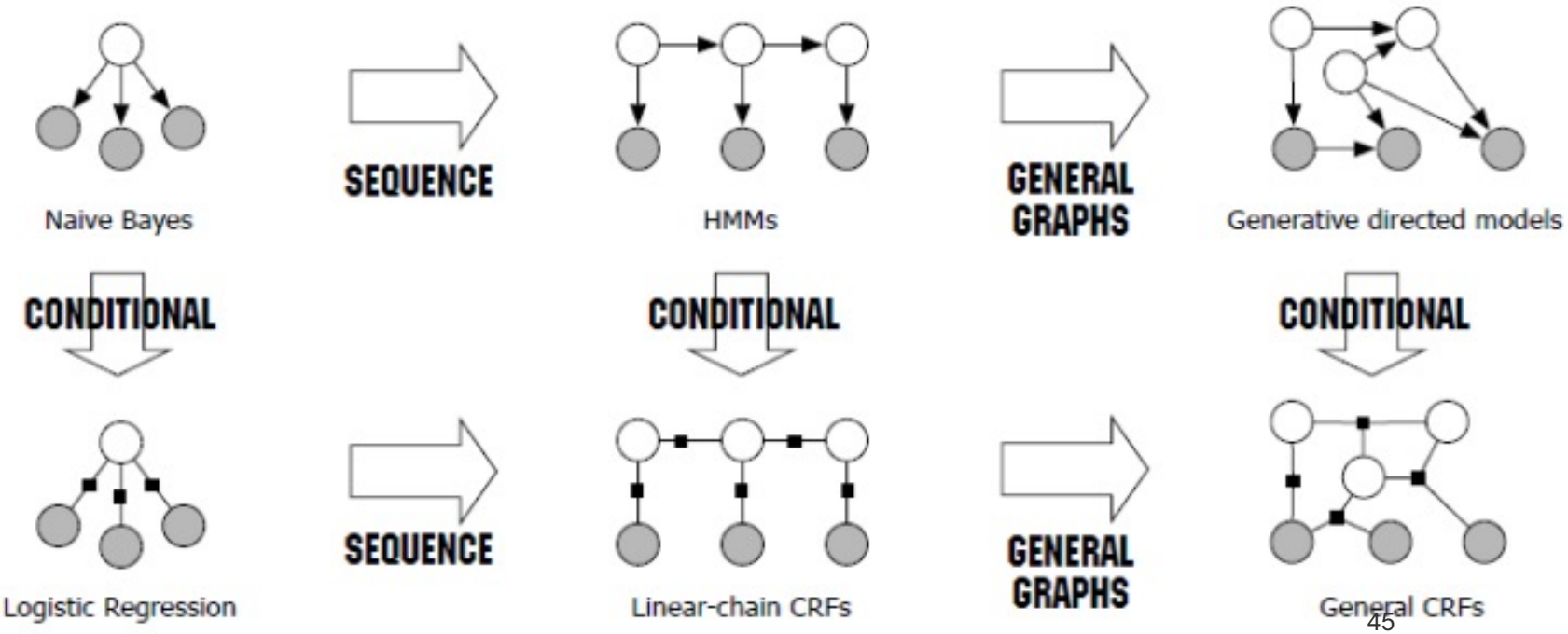


ResNet
152 layers



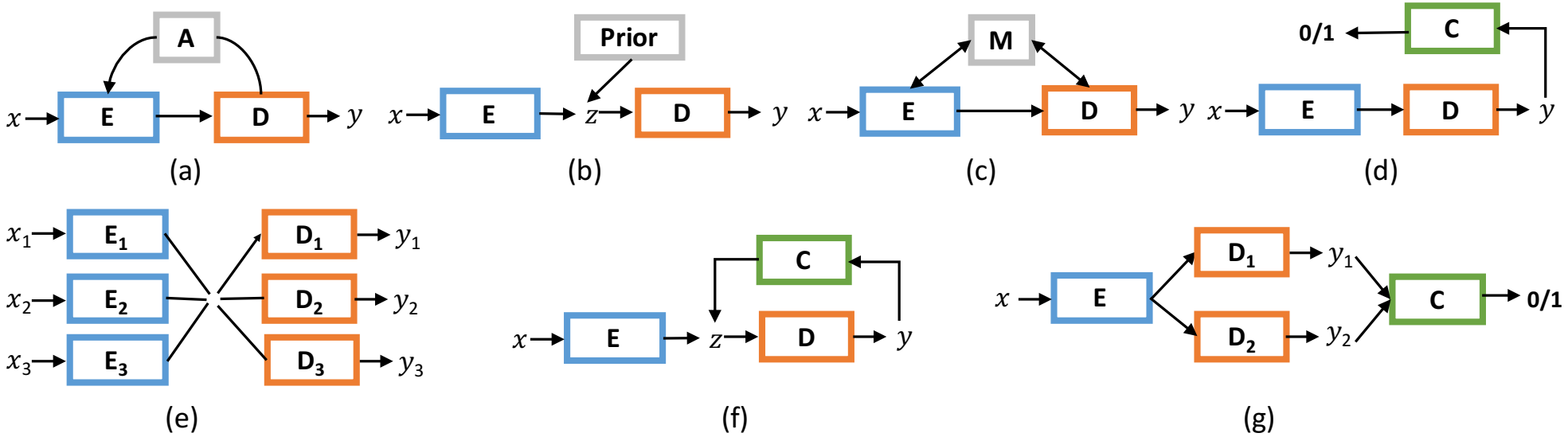
Architecture

- Neural network design
- Graphical model design
- Compositional



Architecture

- Neural network design
- Graphical model design
- Compositional architectures



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

ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture

decoder

LSTM RNN

Attention RNN

Transformer

...

encoder

classifier

ML Components

Optimization

Loss

Divergence

Experience

Operation

Architecture

decoder

LSTM RNN

Attention RNN

Transformer

...

encoder

classifier

Operation, Loss & Optimization (1): Learning from data instances

Loss

- Experience: data instances $(\boldsymbol{x}^*, \boldsymbol{y}^*)$
- Divergence: cross entropy
- Reduce to MLE:

$$\mathcal{L}_{\text{MLE}} = -\log p_\theta(\boldsymbol{y}^* \mid \boldsymbol{x}^*) = -\prod_{t=1}^{T} p_\theta(y_t^* \mid \boldsymbol{y}_{1:t-1}^*, \boldsymbol{x}^*)$$

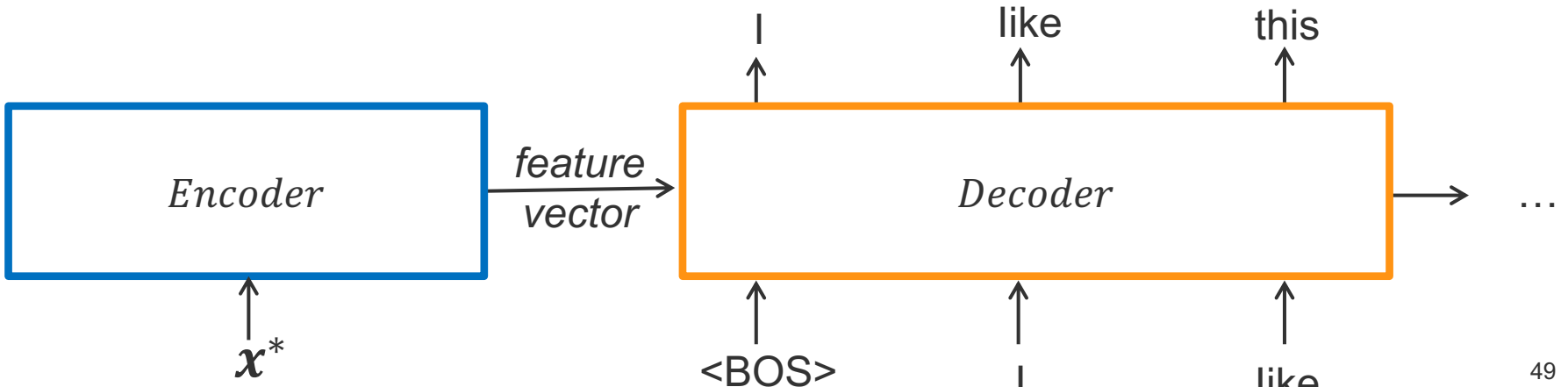
Optimization

SGD

Operation

Teacher-forcing decoding:

For every step t, feeds in the previous ground-truth tokens $y_{1:t-1}^*$ to decode next step



Operation, Loss & Optimization (2): Learning from adversaries + data instances

Loss

- Experience:
  - data instances $(\boldsymbol{x}^*, \boldsymbol{y}^*)$
  - adversary (discriminator)
- Divergence: JSD, f-Div., W-dist.

Optimization

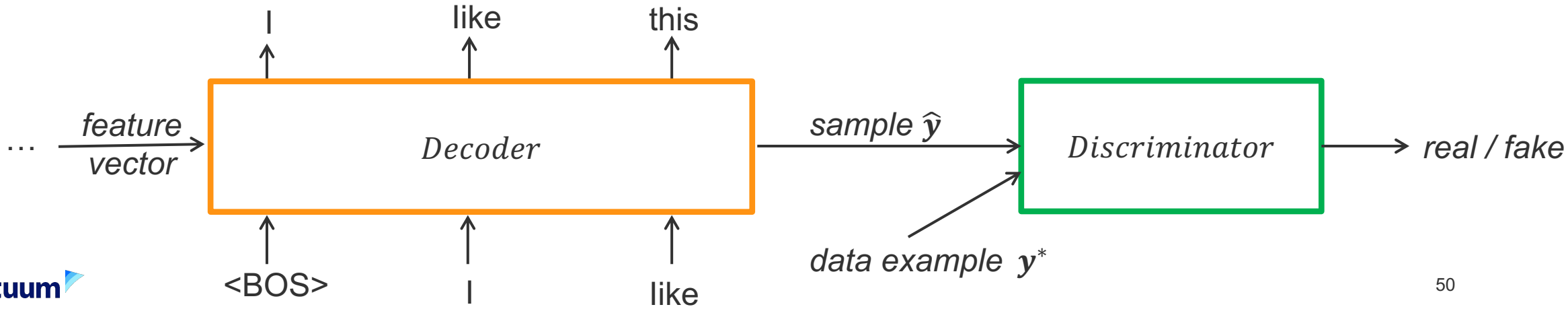
PFD, convex duality

Operation

Gumbel-softmax decoding:

Uses a differentiable approximation of sample $\hat{\boldsymbol{y}}$ for gradient backpropagation

$$\frac{\partial \mathcal{L}(\hat{\boldsymbol{y}})}{\partial \theta} = \frac{\partial \mathcal{L}(\hat{\boldsymbol{y}})}{\partial \hat{\boldsymbol{y}}} \frac{\partial \hat{\boldsymbol{y}}}{\partial \theta}$$



Operation, Loss & Optimization (3): Learning from reward + data instances

Loss

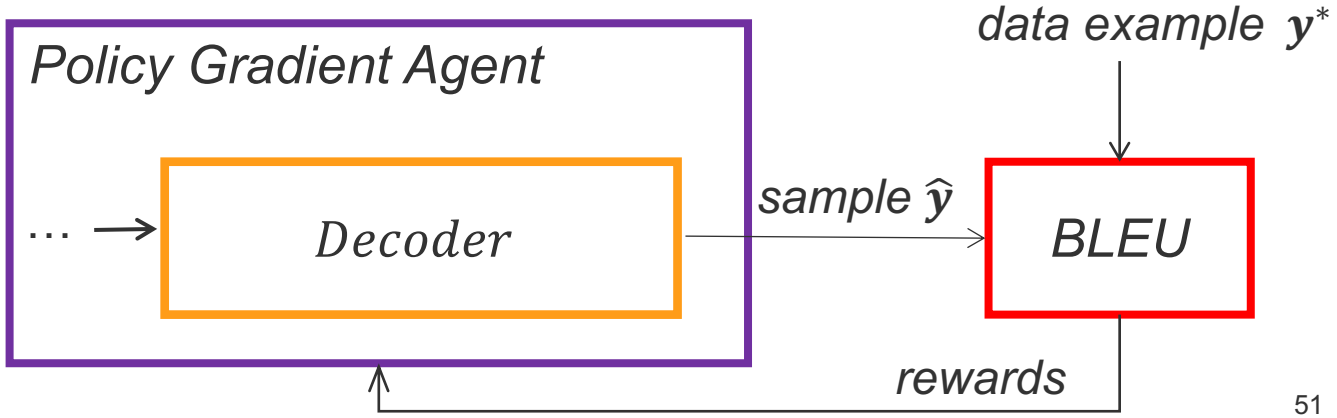
- Experience:
  - data instances $(\boldsymbol{x}^*, \boldsymbol{y}^*)$
  - reward metric, e.g., BLEU
- Divergence: cross entropy

Operation

- Greedy decoding
- Sampling decoding
- Beam search decoding
- Top-k / Top-p decoding
- …

Optimization

EM



Holistic View

Optimization

Loss

Divergence

Experience

Operation

Architecture

SGD

EM

PFD

Episodic RL

…

Cross Ent.

JS Div.

f-Div.

W Dist.

…

Data instances

Reward

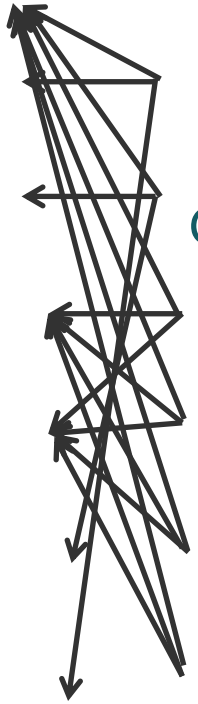
Adversarial model

Data + Reward

Rules

Adversarial + Reward

…



decoding

Teacher-forcing

Gumbel-softmax

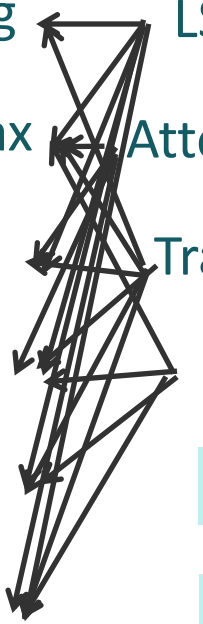
Sample

Greedy

Beam-search

Top-k sample

…



decoder

LSTM RNN

Attention RNN

Transformer

…

encoder

classifier

Holistic View

called as subroutines

uniform interfaces

Optimization

SGD

EM

PFD

Episodic RL

…

Loss

Divergence

Cross Ent.

JS Div.

f-Div.

W Dist.

…

Experience

Data instances

Reward

Adversarial model

Data + Reward

Rules

Adversarial + Reward

…

Operation

decoding

Teacher-forcing

Sample

Greedy

Gumbel-softmax

Beam-search

Top-k sample

…

Architecture

decoder

LSTM RNN

Attention RNN

Transformer

…

encoder

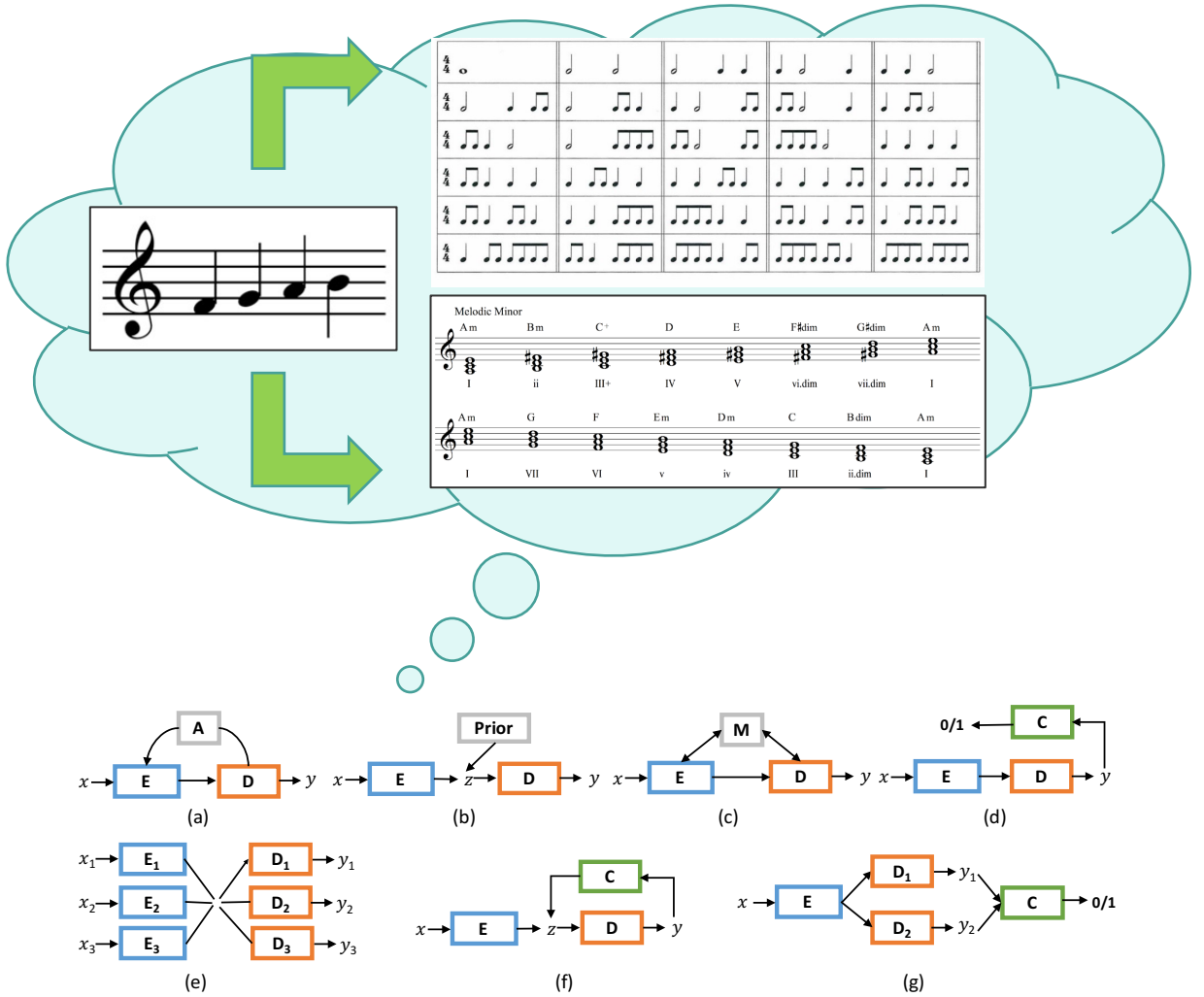
classifier

…

Summary so far: the concept of composable ML

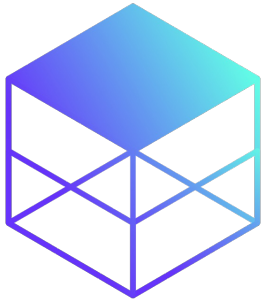
- Composable ML
  - Basic “musical notes” for complex ML systems
  - Structure/rules for combining notes into “chords”/“rhythms” …
  - … and chords/rhythms into compositions
  - (or just think of it as Lego for ML)

- **Next** – Symbolic programming via Petuum Texar open source

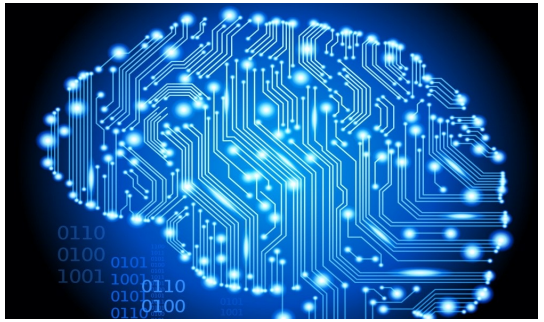


UC San Diego | Mohamed bin Zayed University of Artificial Intelligence | Carnegie Mellon University School of Computer Science | Petuum

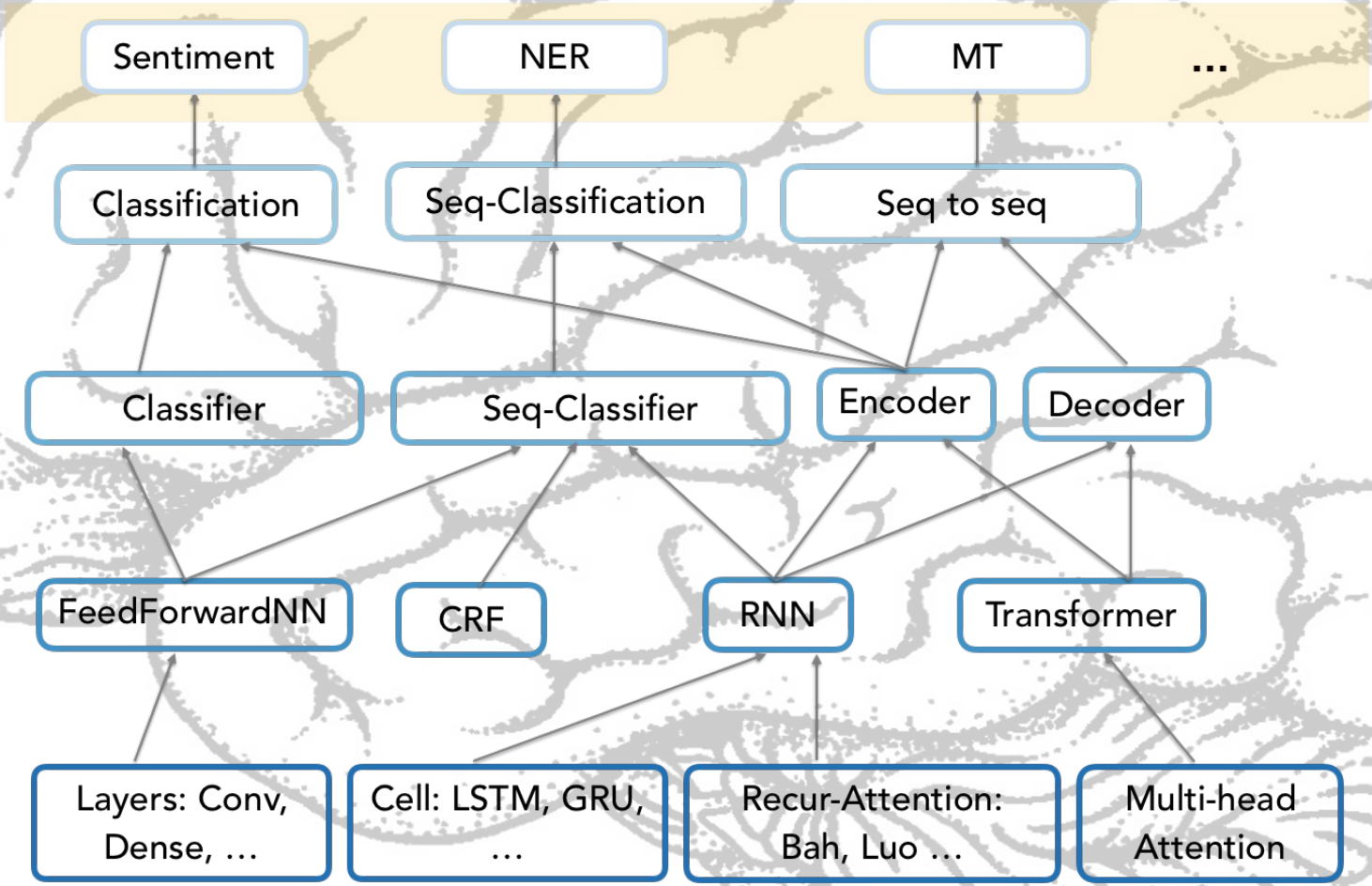
ML Composation with Texar



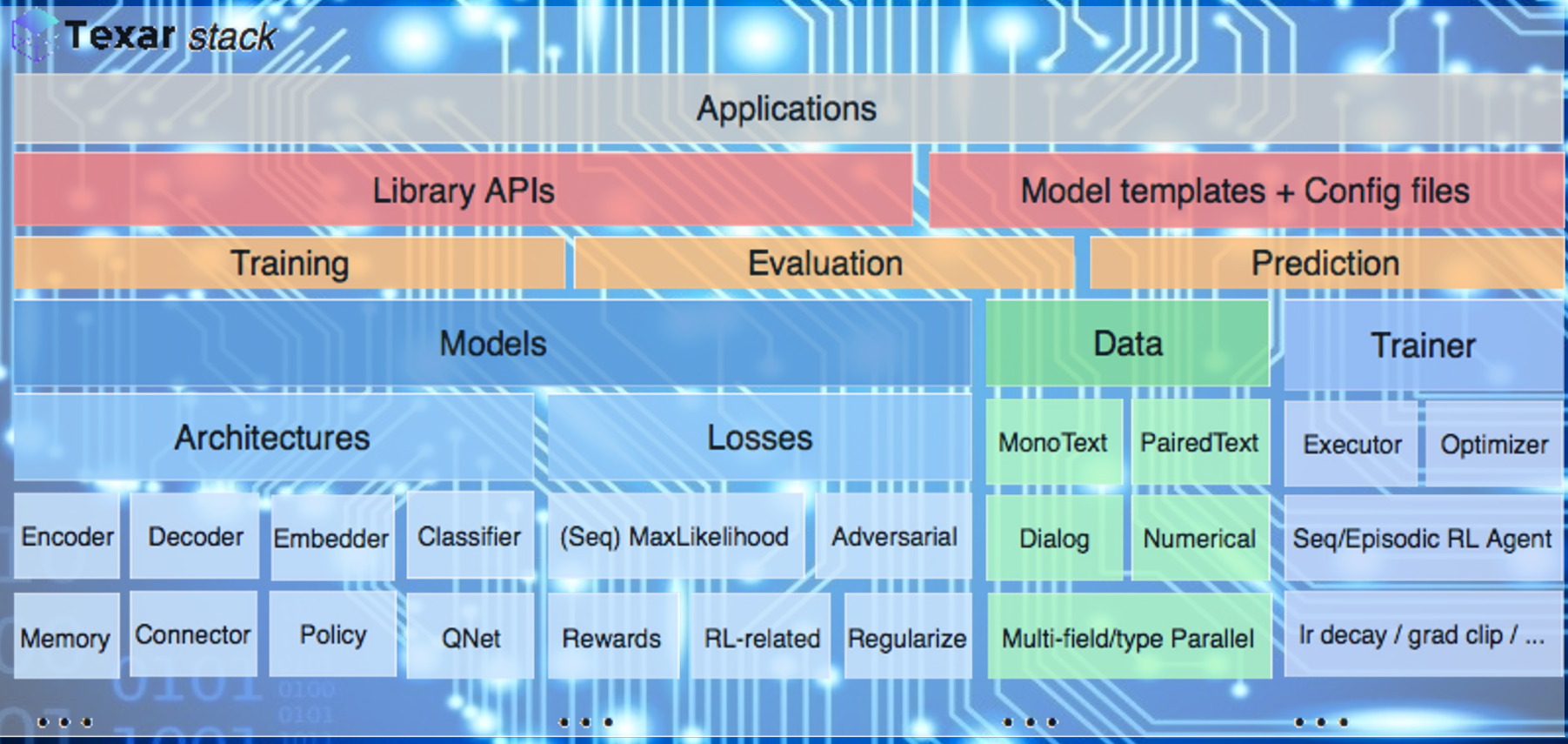
Compose your ML applications like playing building blocks



Expert's Intellectual "View" of Composable ML



Texar Stack – Operationalized "View" of Composable ML



Module Catalog in Texar

Model architecture

- **Encoder**
  - UnidirectionalRNNEncoder
  - BidirectionalRNNEncoder
  - HierarchicalRNNEncoder
  - ConvEncoder
  - TransformerEncoder
  - ...
- **Decoder**
  - BasicRNNDecoder
  - AttentionRNNDecoder
    - BahdananAttn
    - LuongAttn
    - MonotonicAttn
  - TransformerDecoder
  - Greedy/Sample/BeamSearch/ GumbelSoftmax/... Decoding
  - ...
- **Embedder**
  - WordEmbedder (one-hot / soft)
  - PositionEmbedder
    - Parametrized
    - Sinusoids
  - ...
- **Classifier/ Discriminator**
  - RNNClassifier
  - ConvClassifier
  - HierarchicalClassifier
  - ...
- **Connector**
  - MLPTransformer
  - Stochastic
  - ReparameterizedStochastic
  - Concat
  - Forward
  - ...

• • •

Model loss

- **Loss**
  - MLE Loss
    - (Sequence) Cross-entropy
    - ...
  - Adversarial Loss
    - Binary Adversarial Loss
    - ...
  - Rewards
  - ...

Trainer

- **RL Agent**
  - Seq RL Agent
    - Seq Policy Gradient Agent
    - ...
  - Episodic RL Agent
    - Policy Gradient Agent
    - DQN Agent
    - Actor-critic Agent
    - ...
- **Optimization**
  - Optimizer
    - Adam/SGD/...
  - Learning Rate Decay
    - Piecewise/Exp/...
  - ...

Data

- **Data**
  - MonoText
  - PairedText
  - MultiAligned
  - Dialog
  - Numerical
  - ...

• • •

Texar Highlights

Modularized

Assembles any complex model like playing building blocks

Versatile

Supports a large variety of models/algorithms/ applications …

Extensible

Allows to plug in any customized or external modules

$\min_\theta \mathcal{L}(\theta, \mathcal{E})$

Running Example: Machine Translation (MT)

cleaning, tokenizing, truncating, …

source.dat

I like this movie.

Lovely and poignant

Insanely hilarious!

…

target.dat

Ich mag diesen film.

Schön und ergreifend

Wahnsinnig witzig!

…

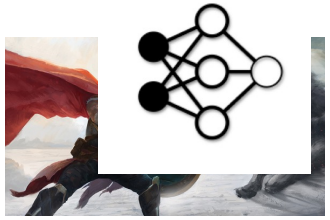
data instances

BLEU

Semantic ROUGE

score

reward



Adversaries

experiences

model architecture

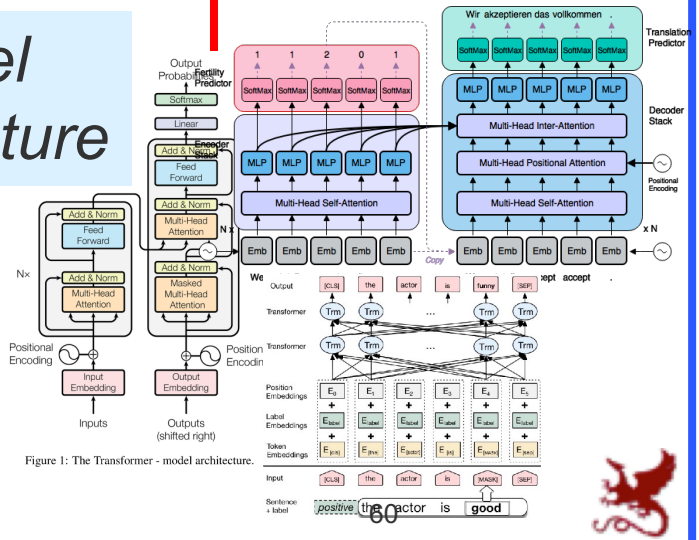


Figure 1: The Transformer - model architecture.

training

$$\min_{q,\theta} \; -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

evaluation post-processing

…

Running Example: MT - Learning from data instances

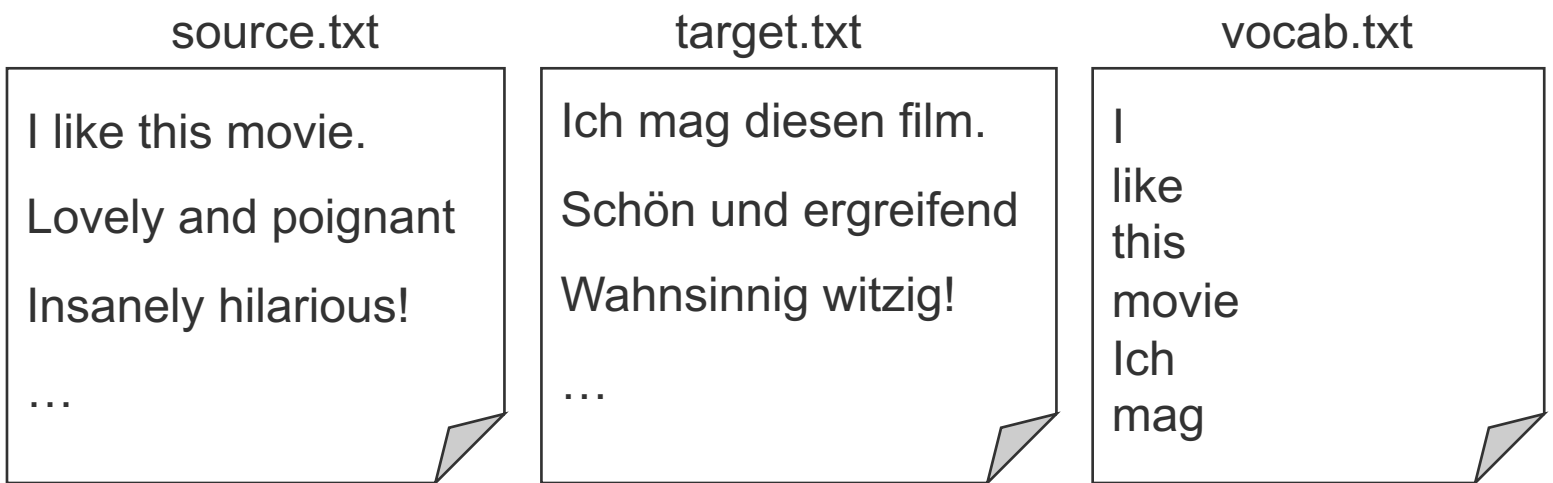
- Input: a source sentence:

  I like this movie.

- Output: a target sentence:

  Ich mag diesen film.

- Dataset:



Running Example: MT - Learning from data instances

Running Example: MT - Learning from data instances

Data

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

YAML config files

```
data_hparams:
    batch_size: 64
    num_epochs: 10
    shuffle: True
    source_dataset:
        files: 'source.txt'
        vocab_file: 'vocab.txt'
        max_seq_length: 100
        bos_token: '<BOS>'
        eos_token: '<EOS>'
    target_dataset:
        …
…
```

Running Example: MT - Learning from data instances

Data

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

YAML config files

Architecture & Inference

Running Example: MT - Learning from data instances

Data

Architecture & Inference

```
# Read data
dataset = PairedTextData(data_hparams)
batch = DataIterator(dataset).get_next()
# Encode
embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
```

YAML config files

```
embedder_hparams:
  embedding_dim: 256
  dropout_rate: 0.9
  regularization: 'L1L2'
…
```

Running Example: MT - Learning from data instances

Data

Architecture & Inference

```
# 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)
```

YAML config files

```
encoder_hparams:
    num_blocks: 16
    num_heads: 8
    hidden_dim: 256
    output_dim: 128
    dropout_rate: 0.8
…
```

Running Example: MT - Learning from data instances

Data

Architecture & Inference

```
1  # Read data
2  dataset = PairedTextData(data_hparams)
3  batch = DataIterator(dataset).get_next()
4  # Encode
5  embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6  encoder = TransformerEncoder(hparams=encoder_hparams)
7  enc_outputs = encoder(embedder(batch['source_text_ids']),
8                        batch['source_length'])
```

Running Example: MT - Learning from data instances

Data

Architecture & Inference

```
1  # Read data
2  dataset = PairedTextData(data_hparams)
3  batch = DataIterator(dataset).get_next()
4  # Encode
5  embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6  encoder = TransformerEncoder(hparams=encoder_hparams)
7  enc_outputs = encoder(embedder(batch['source_text_ids']),
8                        batch['source_length'])
9  # Build decoder
10 decoder = AttentionRNNDecoder(memory=enc_outputs,
11                               hparams=decoder_hparams)
12 # Maximum Likelihood Estimation
13 ## Teacher-forcing decoding
14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
15                              inputs=embedder(batch['target_text_ids']),
16                              seq_length=batch['target_length']-1)
```


Running Example: MT - Learning from data instances

Data

Architecture & Inference

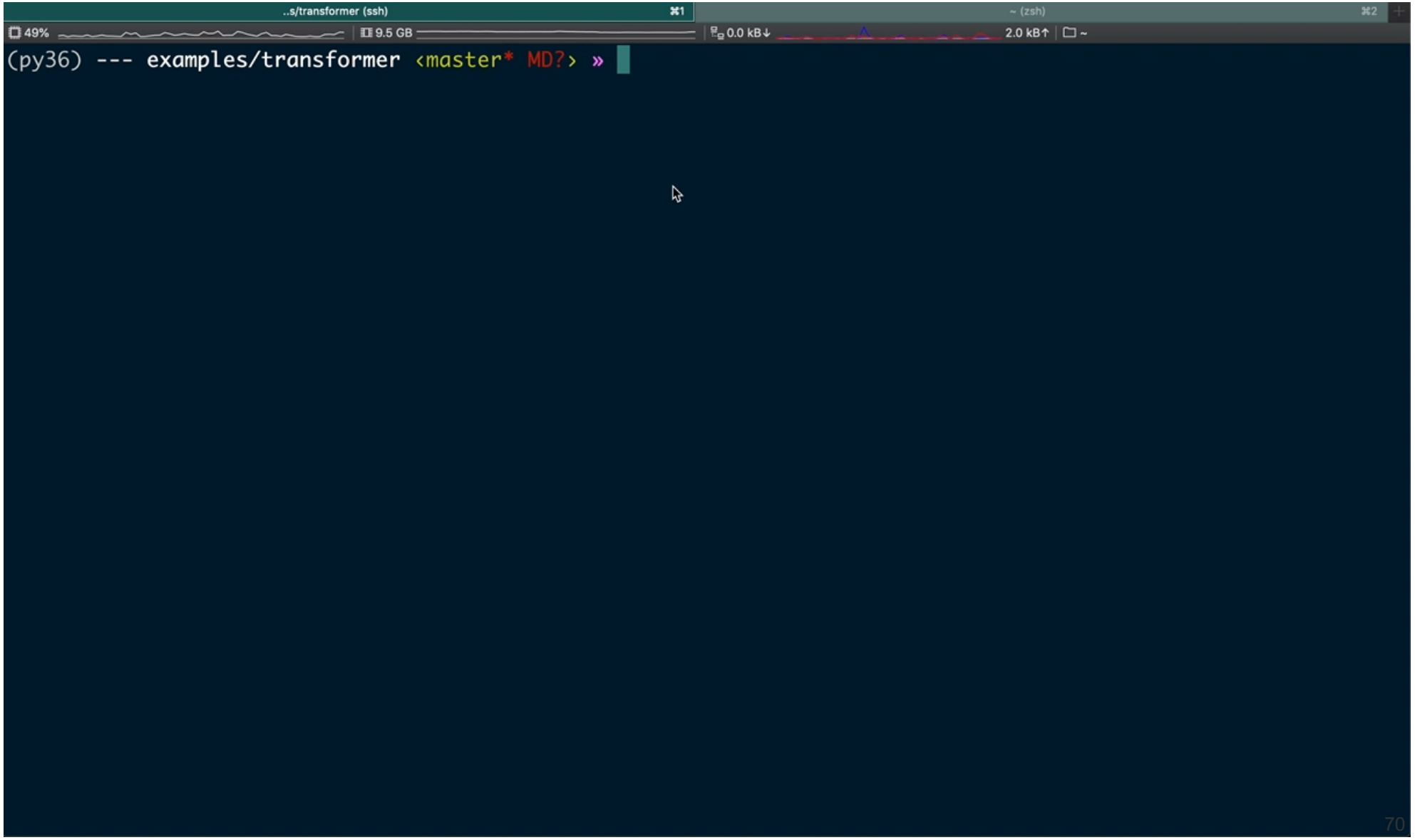
Learning Loss

```
1  # Read data
2  dataset = PairedTextData(data_hparams)
3  batch = DataIterator(dataset).get_next()
4  # Encode
5  embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6  encoder = TransformerEncoder(hparams=encoder_hparams)
7  enc_outputs = encoder(embedder(batch['source_text_ids']),
8                        batch['source_length'])
9  # Build decoder
10 decoder = AttentionRNNDecoder(memory=enc_outputs,
11                                hparams=decoder_hparams)
12 # Maximum Likelihood Estimation
13 ## Teacher-forcing decoding
14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
15                              inputs=embedder(batch['target_text_ids']),
16                              seq_length=batch['target_length']-1)
17 ## Cross-entropy loss
18 loss = sequence_sparse_softmax_cross_entropy(
19    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
20
```

UC San Diego

MOHAMED BIN ZAYED UNIVERSITY OF ARTIFICIAL INTELLIGENCE

DEMO



Petuum

Running Example: MT - Learning from data instances

Data

Architecture & Inference

Learning Loss

```
1  # Read data
2  dataset = PairedTextData(data_hparams)
3  batch = DataIterator(dataset).get_next()
4  # Encode
5  embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
6  encoder = TransformerEncoder(hparams=encoder_hparams)
7  enc_outputs = encoder(embedder(batch['source_text_ids']),
8                        batch['source_length'])
9  # Build decoder
10 decoder = AttentionRNNDecoder(memory=enc_outputs,
11                                hparams=decoder_hparams)
12 # Maximum Likelihood Estimation
13 ## Teacher-forcing decoding
14 outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
15                              inputs=embedder(batch['target_text_ids']),
16                              seq_length=batch['target_length']-1)
17 ## Cross-entropy loss
18 loss = sequence_sparse_softmax_cross_entropy(
19    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
20
```

Maximum likelihood Estimation

Switching to using other experiences

Data

Architecture & Inference

Learning Loss

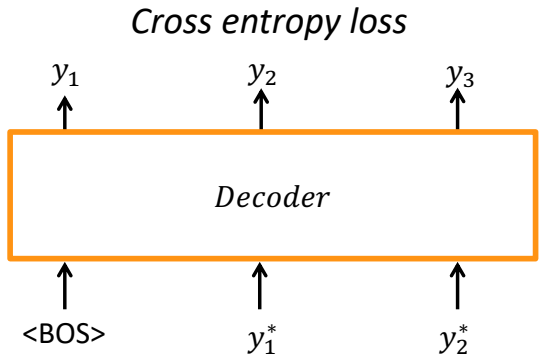
```
# Read data
dataset = PairedTextData(data_hparams)
batch = DataIterator(dataset).get_next()
# Encode
embedder = WordEmbedder(dataset.vocab.size, hparams=embedder_hparams)
encoder = TransformerEncoder(hparams=encoder_hparams)
enc_outputs = encoder(embedder(batch['source_text_ids']),
                      batch['source_length'])
# Build decoder
decoder = AttentionRNNDecoder(memory=enc_outputs,
                              hparams=decoder_hparams)
# Maximum Likelihood Estimation
## Teacher-forcing decoding
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
                             inputs=embedder(batch['target_text_ids']),
                             seq_length=batch['target_length']-1)
## Cross-entropy loss
loss = sequence_sparse_softmax_cross_entropy(
    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)

```

Keep unchanged

Maximum likelihood Estimation

Switching to using reward + data instances

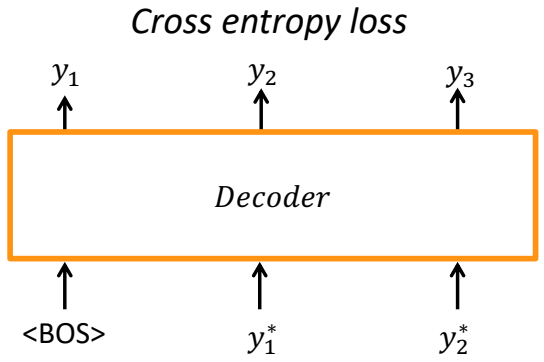
- Maximum likelihood




```
# Teacher-forcing decoding
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
                                        inputs=embedder(batch['target_text_ids']),
                                        seq_length=batch['target_length']-1)
# Cross-entropy loss
loss = sequence_sparse_softmax_cross_entropy(
   labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
```

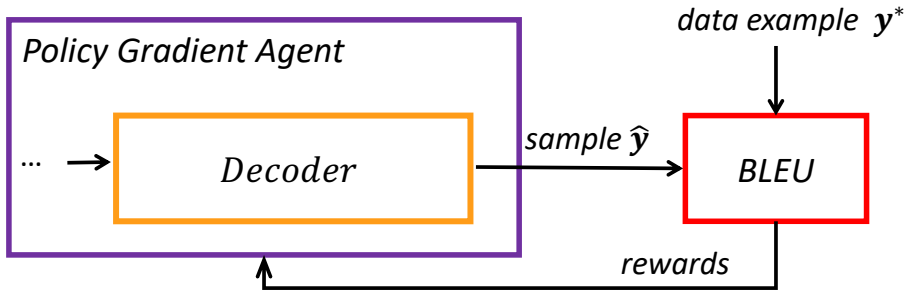
Switching to using reward + data instances

- Maximum likelihood




```
# Teacher-forcing decoding
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
                             inputs=embedder(batch['target_text_ids']),
                             seq_length=batch['target_length']-1)
# Cross-entropy loss
loss = sequence_sparse_softmax_cross_entropy(
    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
```

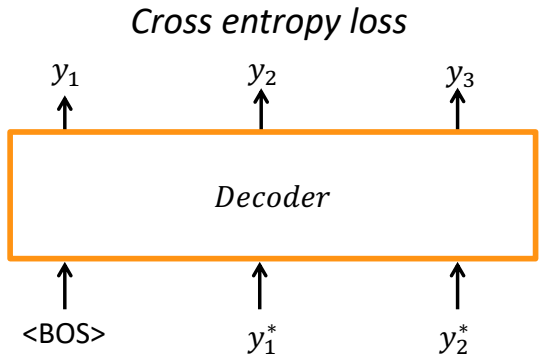
- Reinforcement learning




```
# Random sample decoding
outputs, length, _ = decoder(decoding_strategy='random_sample',
                             start_tokens=[BOS]*batch_size, end_token=EOS,
                             embedding=embedder)
# Policy gradient agent for learning
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)  # Reward
    agent.observe(rewards)
```

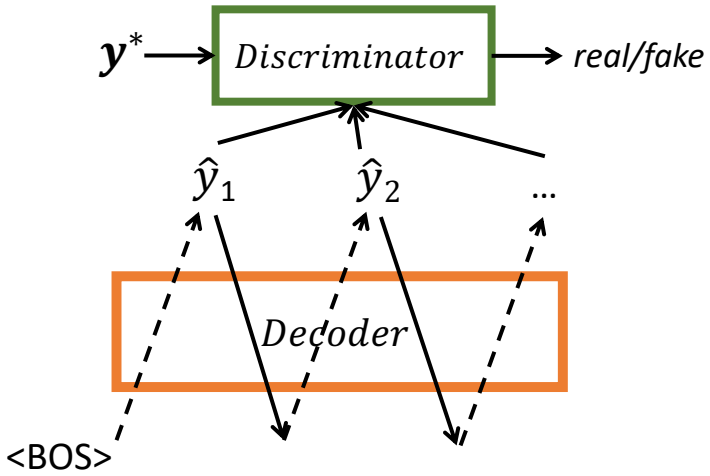
Switching to using adversary + data instances

- Maximum likelihood




```
# Teacher-forcing decoding
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
                             inputs=embedder(batch['target_text_ids']),
                             seq_length=batch['target_length']-1)
# Cross-entropy loss
loss = sequence_sparse_softmax_cross_entropy(
    labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)
```

- Adversarial learning



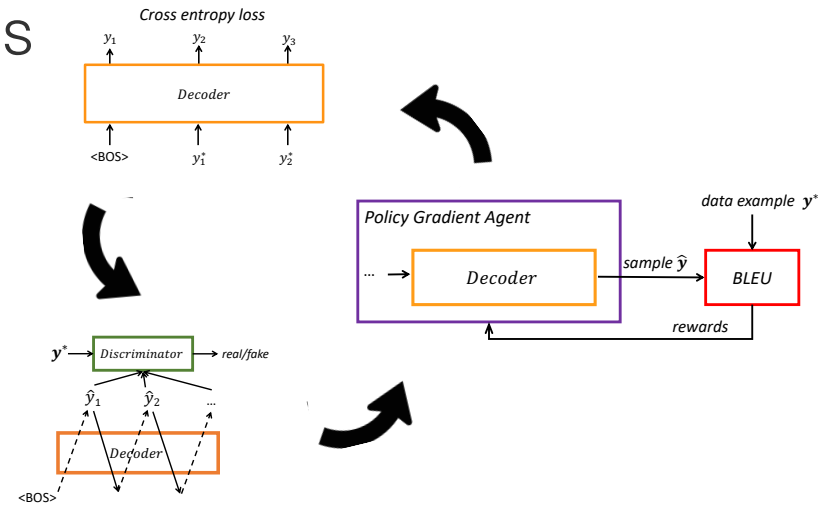

```
# Gumbel-softmax decoding
outputs, _, _ = decoder(decoding_strategy= ='gumbel-softmax',
                        start_tokens=[BOS]*batch_size, end_token=EOS,
                        embedding=embedder)

discriminator = Conv1DClassifier(hparams=conv_hparams)
# Binary adversarial loss
G_loss, D_loss = binary_adversarial_losses(
    embedder(batch['target_text_ids'][:, 1:]),
    embedder(soft_ids=softmax(outputs.logits)),
    discriminator)
```

Summary of MT in Texar

- Highly modularized programming
  - Experience, arch, operation, loss, optimization
  - Intuitive conceptual-level APIs

```
# 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'])
# Build decoder
decoder = AttentionRNNDecoder(memory=enc_outputs,
                              hparams=decoder_hparams)
# Maximum Likelihood Estimation
## Teacher-forcing decoding
outputs, length, _ = decoder(decoding_strategy='teacher-forcing',
                             inputs=embedder(batch['target_text_ids']),
                             seq_length=batch['target_length']-1)
## Cross-entropy loss
loss = sequence_sparse_softmax_cross_entropy(
  labels=batch['target_text_ids'][:,1:], logits=outputs.logits, seq_length=length)

```

- Easy switch between different forms of experiences
  - Plug in & out modules
  - No changes to irrelevant parts



Applications of Texar

Many products built on Texar

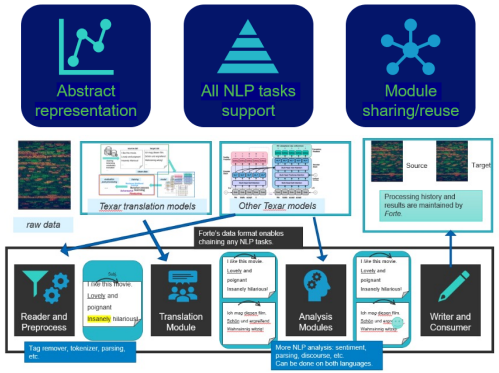
- FORTE – templates for larger complex NLP applications

- Chest X-Ray report writer
- Medical Registry report writer
- ICD coding system

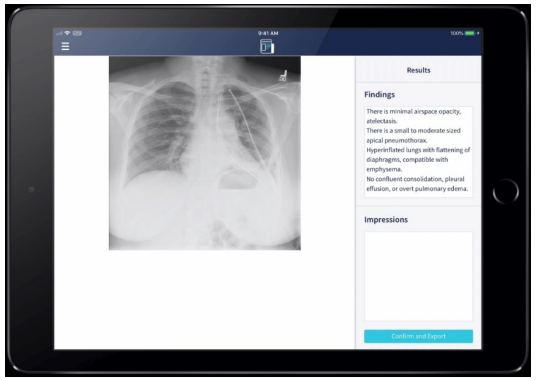
- Financial knowledge base builder
- Financial summary/report writer

- Multi-Lingual Cognitive Chat Bots
  - For Call Center Support
  - For Retail In-Store Assistance

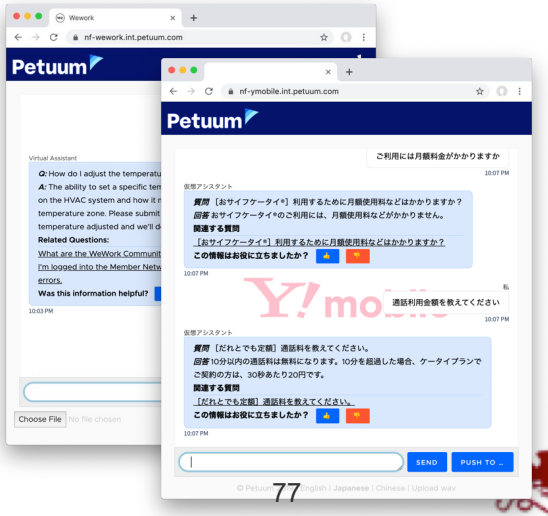
FORTE



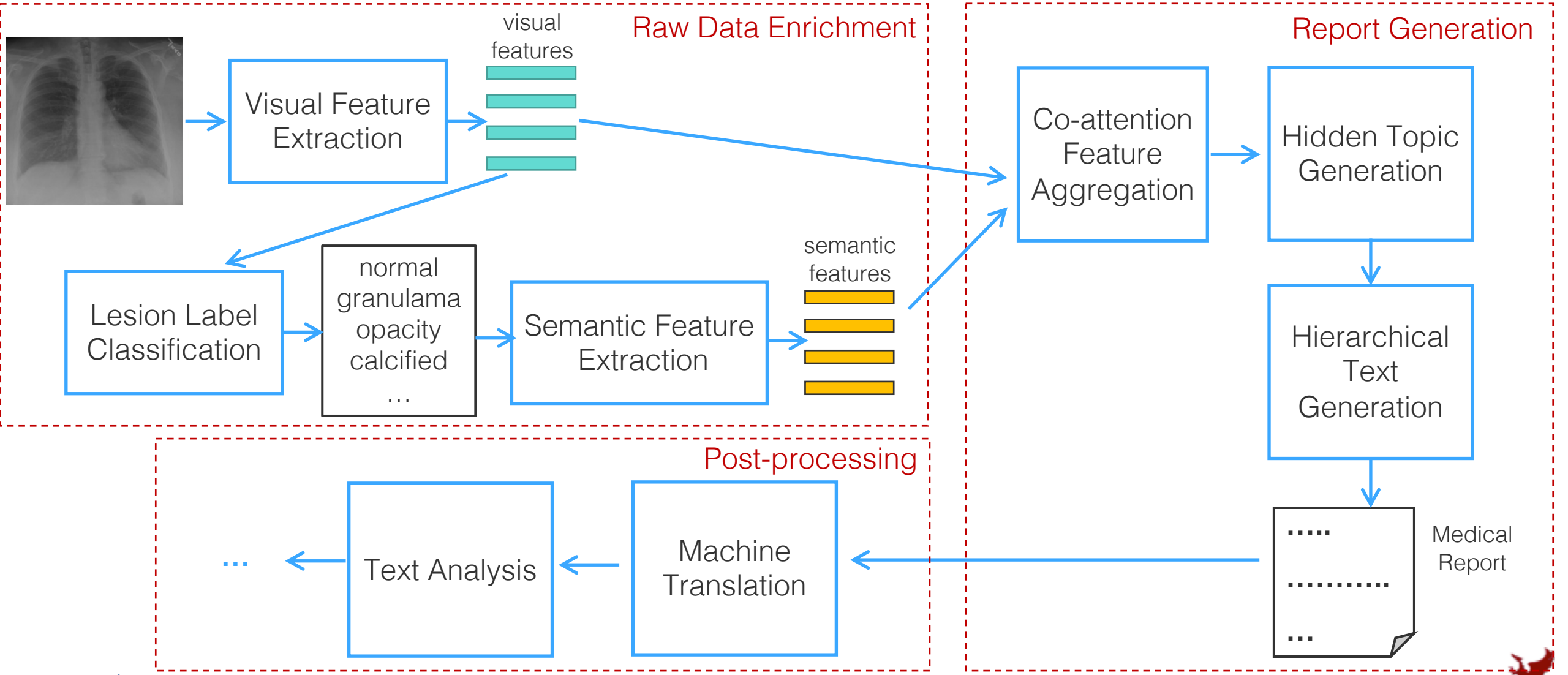
Chest X-Ray Report Writer



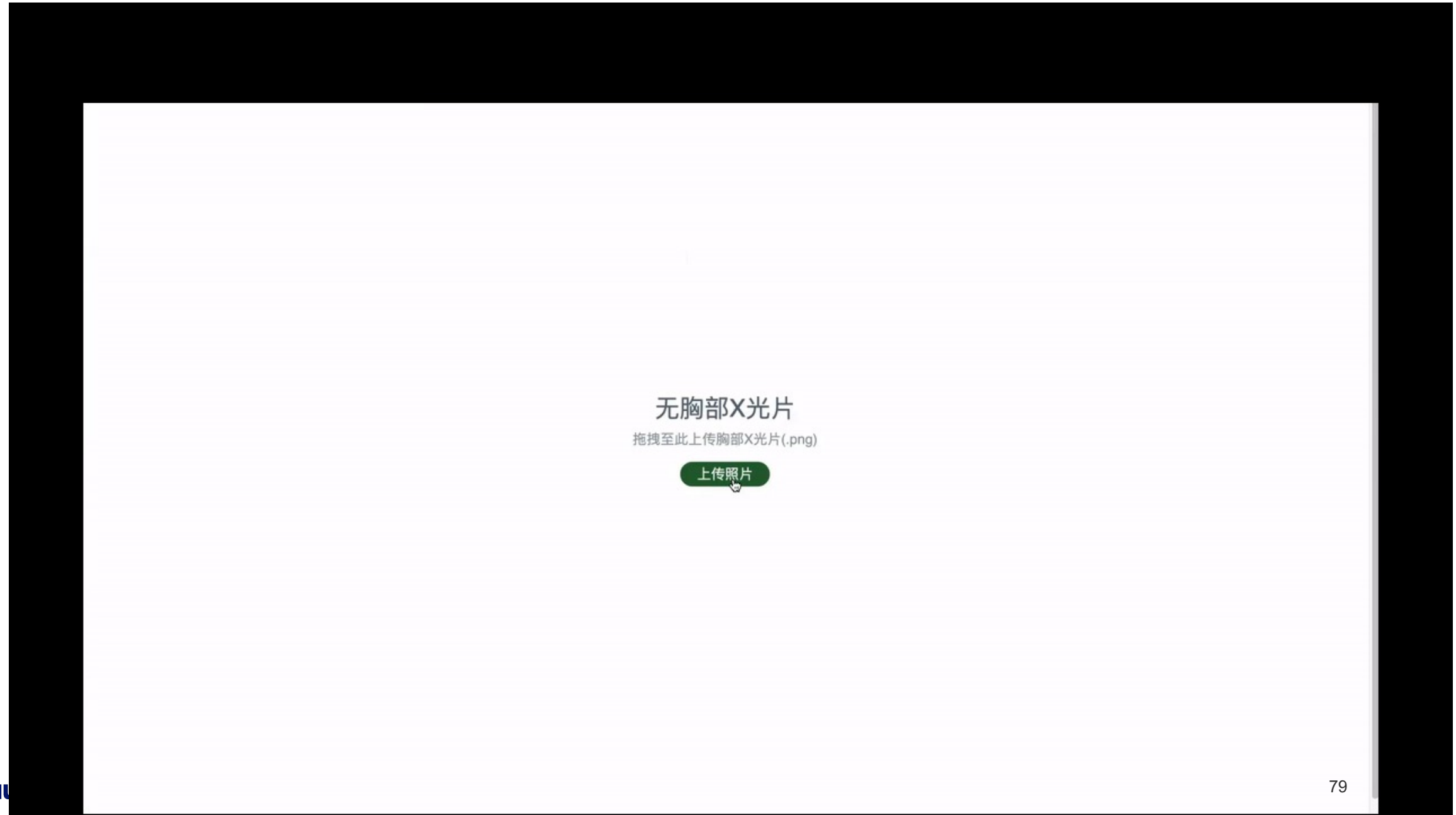
Multi-Lingual Cognitive Chat Bots



Example: Chest X-Ray Report Writer

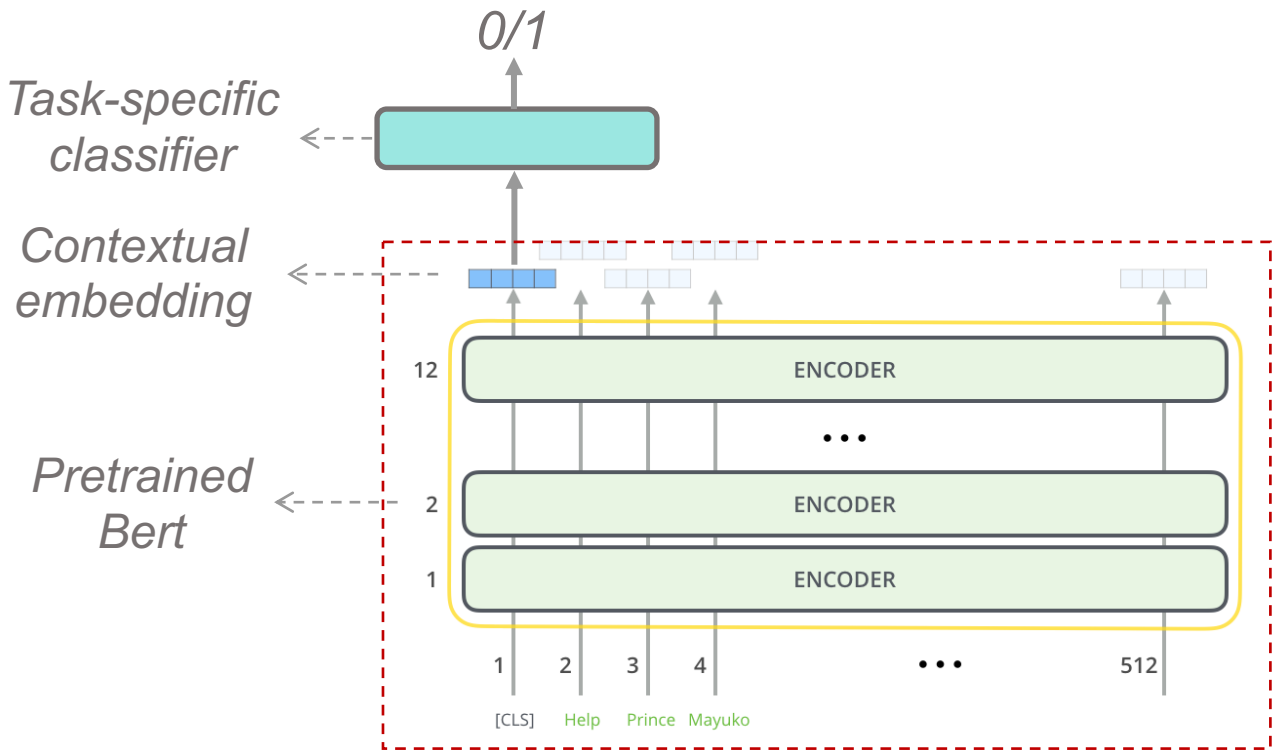


Example: Chest X-Ray Report Writer + Translation



Transfer Learning with Large Pretrained Models Using Texar

- Large neural models pretrained on massive data
  - Elmo, Bert, GPT-2, GPT-3, …
- Pretrained models serve as building blocks to construct downstream models
  - Fine-tune using task-specific experiences



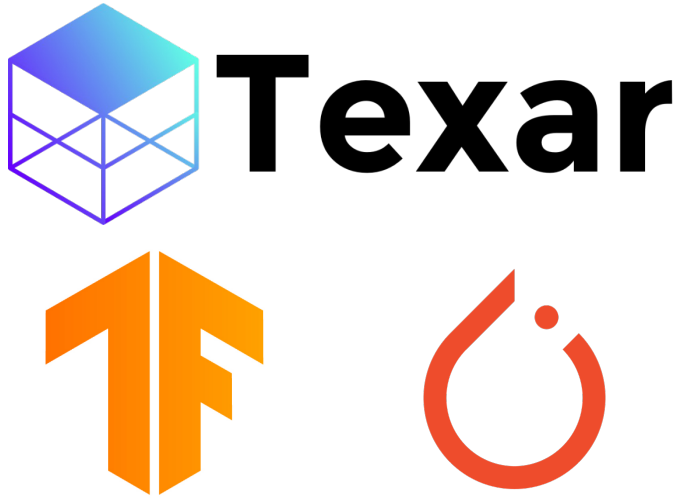

```
model = BertEncoder()
_, features = model(input_ids, input_length, segment_ids)
task_classifier = Conv1DClassifier(hparams={…})
logits, preds = task_classifier(features[:,:1,:])
```

OR

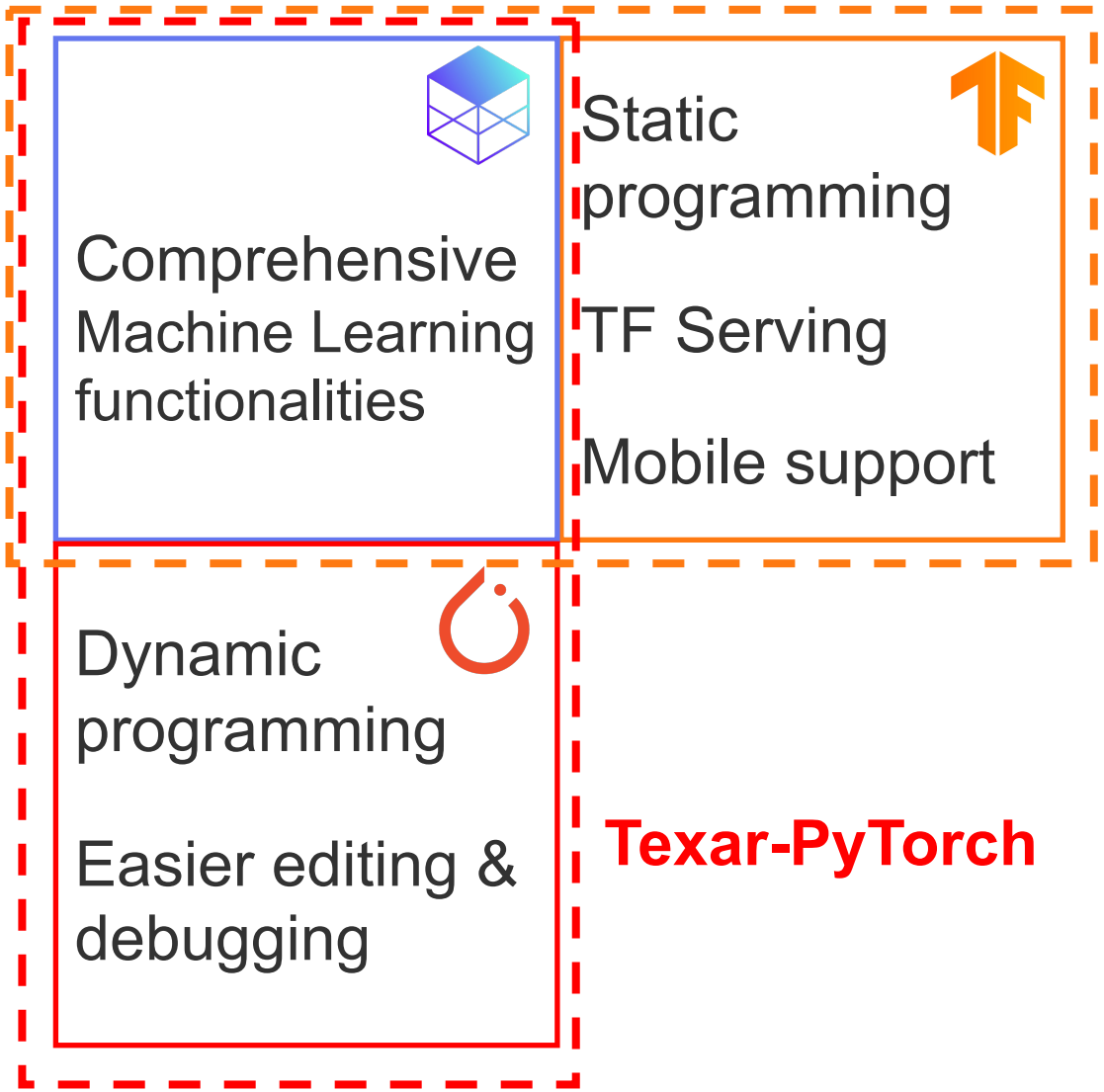
```
task_classifier = BertClassifier(hparams={'clas_strategy': 'cls_time'})
logits, preds = task_classifier(input_ids, input_length, segment_ids)
```

Support of TensorFlow and PyTorch

- Texar is built upon TF and PyTorch
  - Texar-TF & Texar-PyTorch: mostly the same interfaces!
  - Higher-level intuitive APIs without loss of flexibility
  - Lots of ML components ready to use
- Combine the best design of TF and PyTorch
  - TF:
    - Easy and efficient data processing APIs
    - Excellent factorization of ML modules
    - Turnkey model training processor
  - PyTorch:
    - Intuitive programming interfaces
    - Transparent variable scope and sharing to users



Support of TensorFlow and PyTorch



- Texar provides the same comprehensive supports of ML functionalities
- Fully compatible with native TF & PyTorch APIs
  - Get all other useful features of TF or PyTorch with Texar-TF & Texar-PyTorch

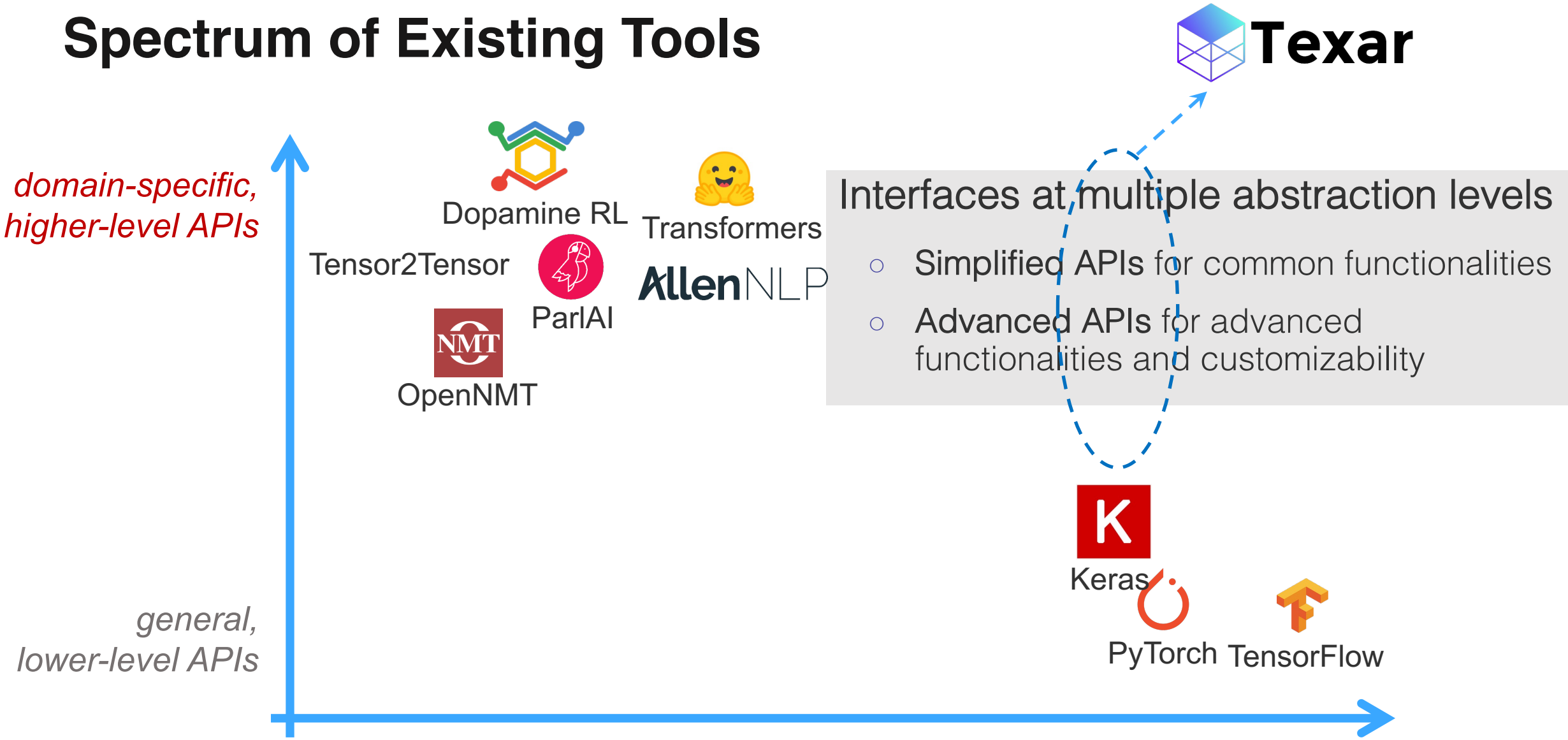
Texar Resources

Texar-TF

Texar-PyTorch

- Website: https://asyml.io
- GitHub (TF version): https://github.com/asyml/texar
- GitHub (PyTorch version): https://github.com/asyml/texar-pytorch
- 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

Spectrum of Existing Tools



Spectrum of Existing Tools

- Symbolic languages for low-level development of ML models
  - Tensorflow, PyTorch, Keras, Jax, …
- High-level, domain-specific APIs
  - Transformers: rich SOTA pretrained transformer models
  - AllenNLP: ready-to-use models for many NLP tasks
  - Dopamine RL: APIs for reinforcement learning
  - …

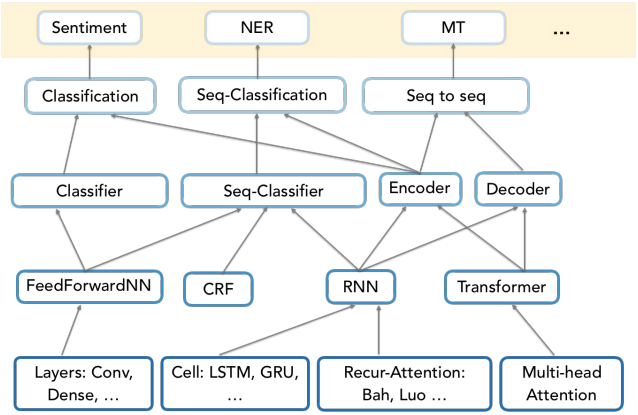
Summary of Part-II

- ML solution design
  - Plugging arbitrary available experiences in learning

$$\min_{q,\theta} \; -\mathbb{E} + \mathbb{D} - \mathbb{H}$$

$$f = w_1 \cdot f(\boldsymbol{x}\,|\,\text{🗄}) + w_2 \cdot f(\boldsymbol{x}\,|\,\text{📖}) + w_3 \cdot f(\boldsymbol{x}\,|\,\text{💲}) + w_4 \cdot f(\boldsymbol{x}\,|\,\text{📣}) + \cdots$$

- Tooling for Compositionality in ML
  - Interfacing and composition between architecture, operation, loss (experience, divergence), optimization
  - ML compositionality with Texar



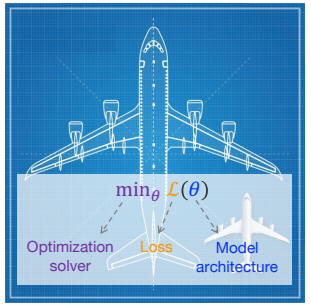
Texar

Schedule

- **Lecture#1:** Theory: The Standard Model of ML

  A blueprint of ML paradigms for ALL experience

  (Jan 19 Thursday, 4:45pm-6:15pm UK Time)



- **Lecture#2:** Tooling: Operationalizing The Standard Model

  Compose your ML solutions like playing Lego

  (Jan 20 Thursday, 1:00pm-2:30pm)

- **Lecture#3:** Computing: Modern infrastructure for productive ML

  Automatic tuning, distributing, and scheduling

  (Jan 20 Thursday, 4:45pm-6:15pm)

Summary

Petuum's open-source collection:
https://github.com/petuum

- Theory: The Standard Equation
  - A blueprint of ML paradigms for ALL experiences $\min_{q,\theta} -\mathbb{E}_{q(\boldsymbol{x},\boldsymbol{y})}\left[f(\boldsymbol{x},\boldsymbol{y})\right] + \alpha\mathbb{D}\left(q(\boldsymbol{x},\boldsymbol{y}), p_\theta(\boldsymbol{x},\boldsymbol{y})\right) - \beta\mathbb{H}(q)$
  - Experience function f can encode different types of experiences
    - Data instances, constraints, informativeness, reward, adversary models, …
  - Turnkey mechanism for learning with ALL experiences
    - Re-use originally specialized algorithms to other contexts
    - Experience compositonality
- Tooling: Operationalizing "Learning with all experiences"
  - ML solution design by plugging arbitrary experiences in learning
  - ML compositionality with Texar

$$f = w_1 \cdot f(\boldsymbol{x}\,|\,\text{🗄}) + w_2 \cdot f(\boldsymbol{x}\,|\,\text{📖}) + w_3 \cdot f(\boldsymbol{x}\,|\,\text{💲}) + w_4 \cdot f(\boldsymbol{x}\,|\,\text{👍}) + \cdots$$

- Computing: Modern infrastructure for productive ML
  - Task/model interoperation
  - Automatic tuning, distributing, and scheduling

Forte AutoDist TUUN AdaptDL

Food for thoughts: How far would this take us?

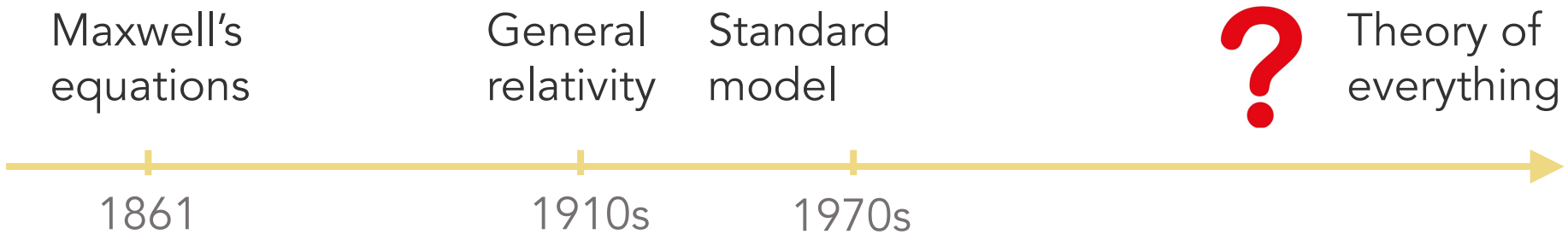
- Physics

It is only slightly overstating the case to say that ***physics is the study of symmetry****.*

-- Phil Anderson (1923-2020), Physicist, Nobel laureate

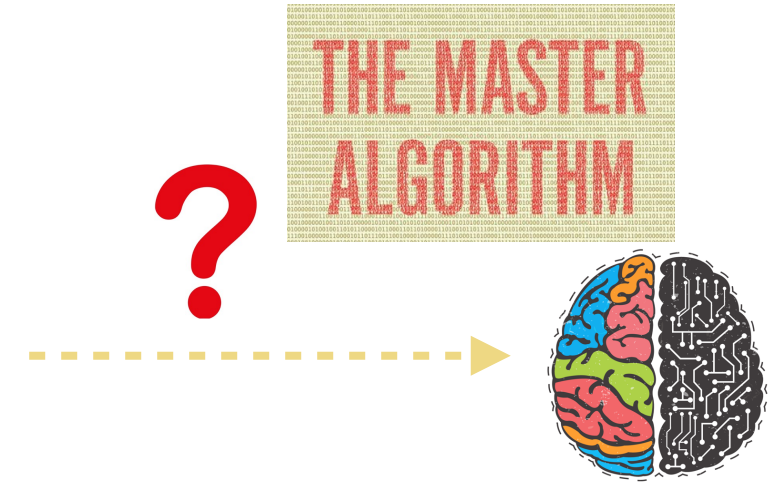
Food for thoughts: How far would this take us?

- Physics



- Machine Learning

  Unified way of thinking
  - ✦ Systematic understanding
  - ✦ Automated solution creation
  - ✦ Improved ML productivity and accessibility



References

[1] Jun Zhu, Ning Chen, and Eric P Xing. 2014. Bayesian inference with posterior regularization and applications to infinite latent SVMs. JMLR(2014).

[2] Jun Zhu and Eric P Xing. 2009. Maximum Entropy Discrimination Markov Networks. JMLR(2009).

[3] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. 2016.Harnessing deep neural networks with logic rules. In ACL.

[4] Zhiting Hu, Haoran Shi, Bowen Tan, Wentao Wang, Zichao Yang, Tiancheng Zhao, Junxian He, Lianhui Qin, Di Wang, Xuezhe Ma, et al. 2019. Texar: A modularized, versatile, and extensible toolkit for text generation. ACL(2019).

[5] Zhiting Hu, Bowen Tan, Russ R Salakhutdinov, Tom M Mitchell, and Eric P Xing. 2019. Learning data manipulation for augmentation and weighting. In NeurIPS.

[6] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In ICML.

[7] Zhiting Hu, Zichao Yang, Ruslan Salakhutdinov, and Eric P Xing. 2018. On Unifying Deep Generative Models. In ICLR.

[8] Zhiting Hu, Zichao Yang, Russ R Salakhutdinov, Lianhui Qin, Xiaodan Liang, Haoye Dong, and Eric P Xing. 2018. Deep generative models with learnable knowledge constraints. In NeurIPS.

References

[9] Hao Zhang, Zeyu Zheng, Shizhen Xu, Wei Dai, Qirong Ho, Xiaodan Liang, Zhiting Hu, Jinliang Wei, Pengtao Xie, Eric Xing. Poseidon: An Efficient Communication Architecture for Distributed Deep Learning on GPU Clusters. ATC 2017

[10] Wei Dai, Yi Zhou, Nanqing Dong, Hao Zhang, Eric P. Xing. Toward Understanding the Impact of Staleness in Distributed Machine Learning. ICLR 2019

[11] Hao Zhang*, Shizhen Xu*, Graham Neubig, Wei Dai, Qirong Ho, Guangwen Yang, and Eric P. Xing. Cavs: A Vertex-centric Programming Interface for Dynamic Neural Networks. ATC 2018

[12] Aurick Qiao, Abutalib Aghayev, Weiren Yu, Haoyang Chen, Qirong Ho, Garth A Gibson, Eric P Xing. Litz: Elastic Framework for High-Performance Distributed Machine Learning. ATC 2018

[13] Kirthevasan Kandasamy, Karun Raju Vysyaraju, Willie Neiswanger, Biswajit Paria, Christopher R Collins, Jeff Schneider, Barnabas Poczos, Eric P Xing. Tuning hyperparameters without grad students: Scalable and robust bayesian optimisation with dragonfly. JMLR 2020

[14] E. P. Xing, Q. Ho, P. Xie, W. Dai, Strategies and Principles of Distributed Machine Learning on Big Data, Engineering, Volume:2, pp179 - 95, 2016.


UC San Diego
MOHAMED BIN ZAYED
UNIVERSITY OF
ARTIFICIAL INTELLIGENCE
Carnegie Mellon University
School of Computer Science
Petuum

Thank You