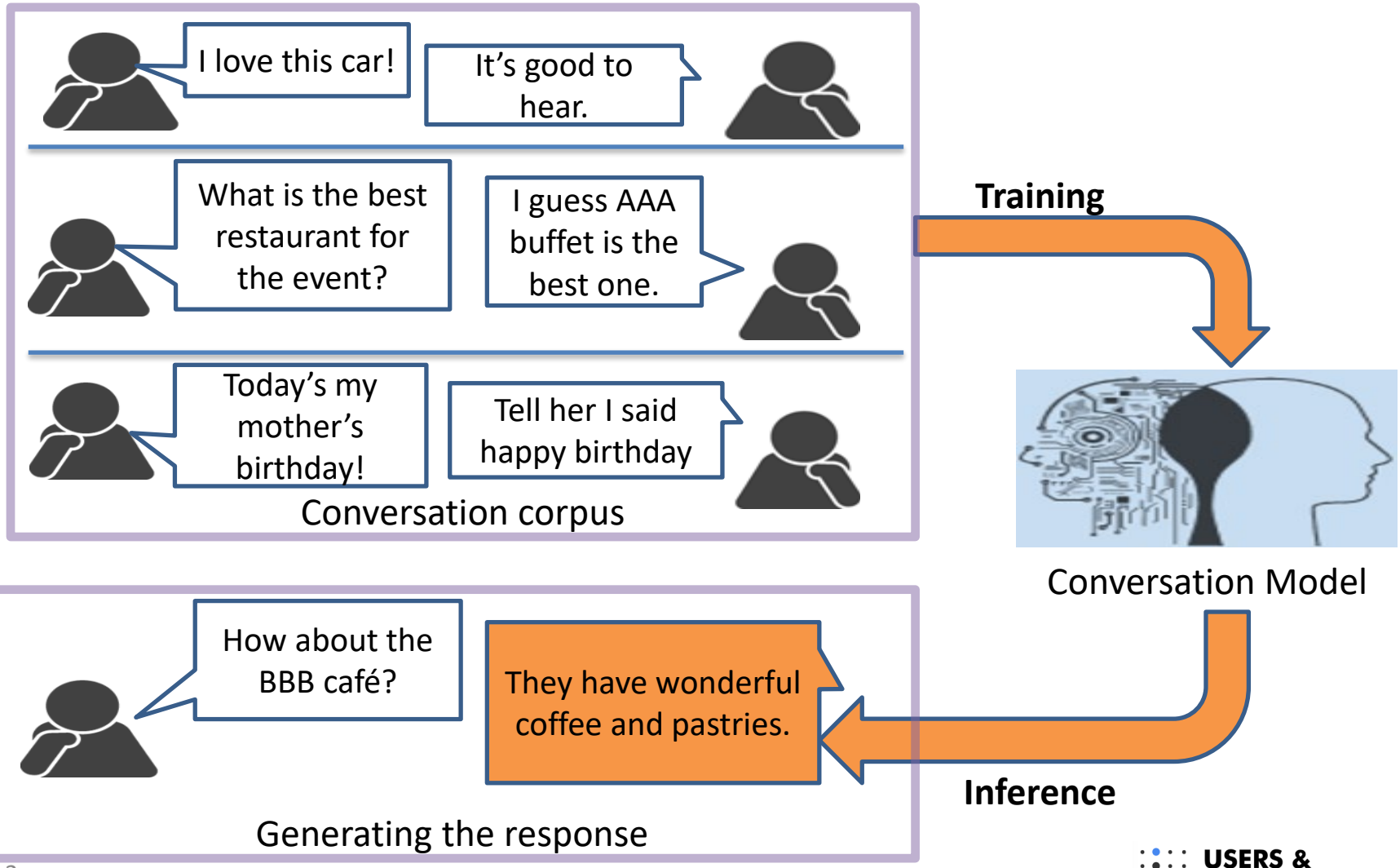
Hyundai Heavy Industries - KAIST AI Conversation Model

JinYeong Bak
jy.bak@kaist.ac.kr
School of Computing, KAIST

USERS & INFORMATION

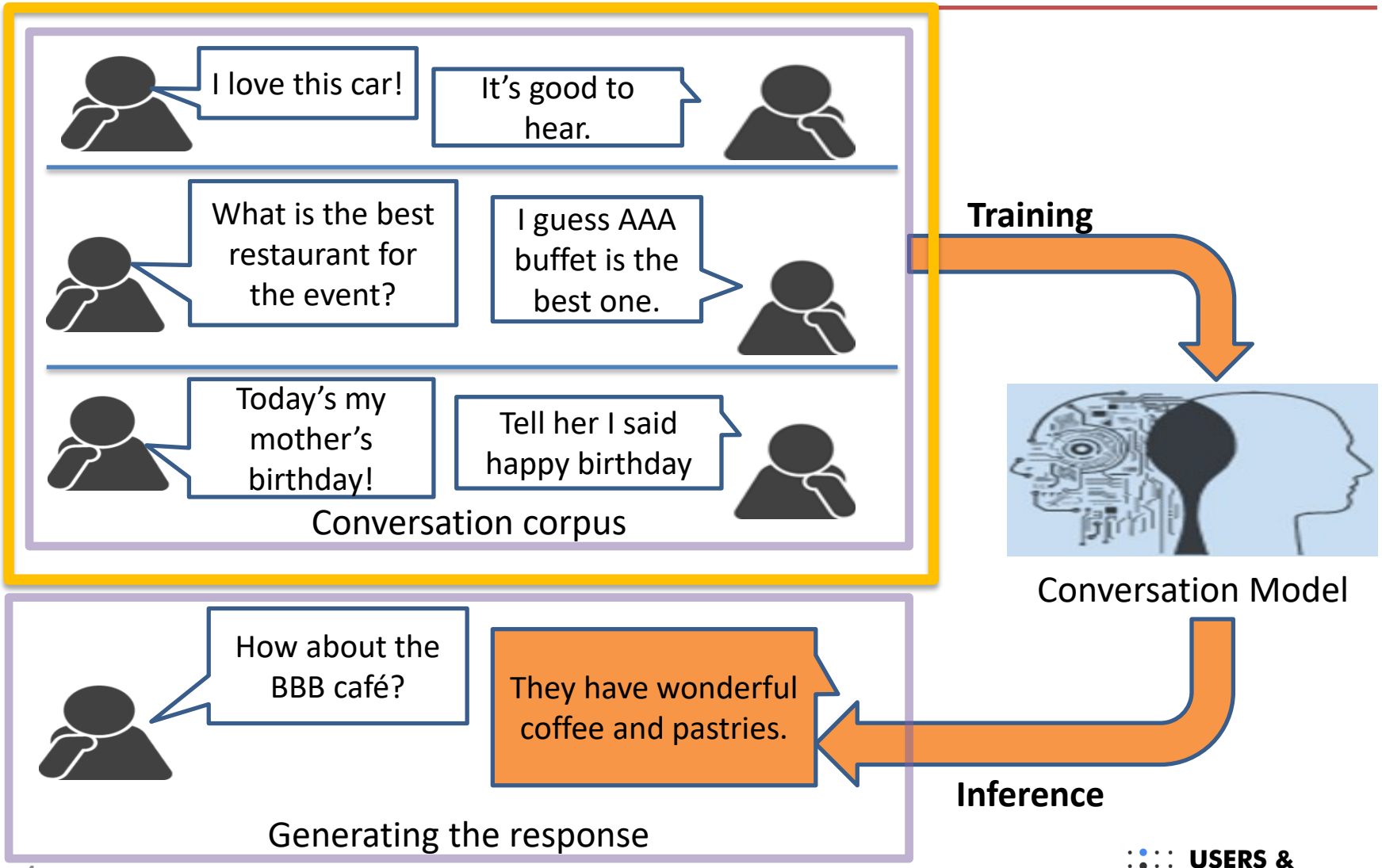
Conversation Model



USERS & INFORMATION

CONVERSATION CORPUS

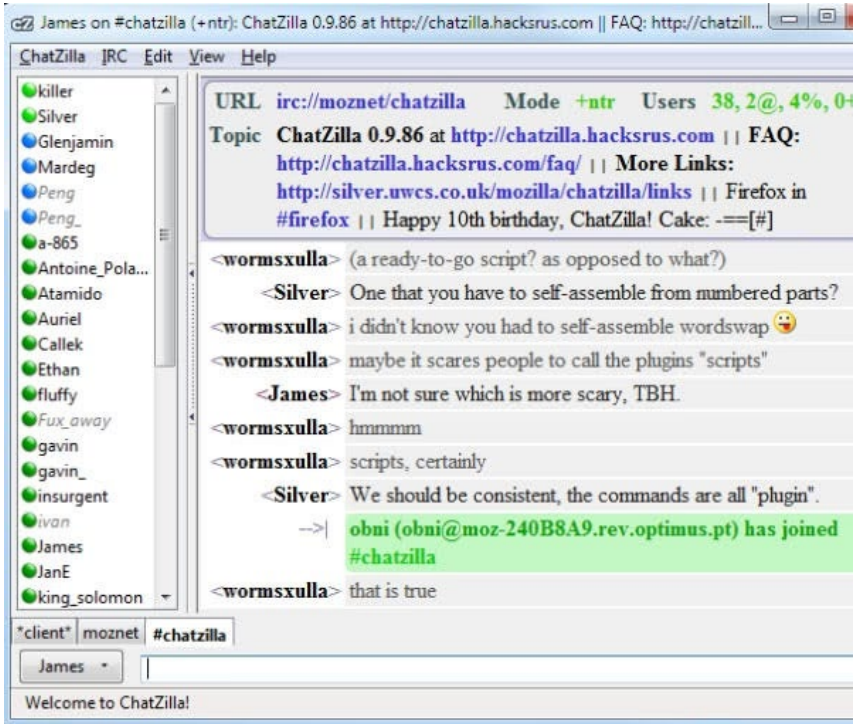
Conversation Model



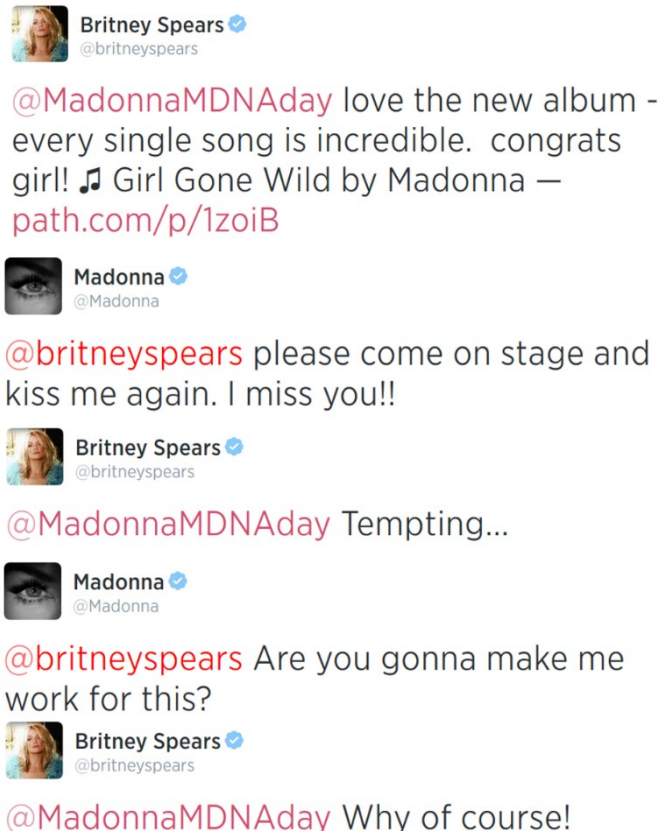
USERS & INFORMATION

Conversation Corpus

IRC



Twitter



USERS & INFORMATION

Conversation Corpus

Movie script

https://youtu.be/fmX2VzsB25s?t=18

Conversation Corpus

Movie script example)

DON CORLEONE:

You look terrible. I want you to eat well, to rest. And spend time with your family. And then, at the end of the month, this big shot will give you the part you want.

JOHNNY:

It's too late. All the contracts have been signed, they're almost ready to shoot.

DON CORLEONE:

I'll make him an offer he can't refuse.

Conversation Corpus

Cornell Movie-Dialogs Corpus

- 617 movies
- 9,035 characters
- 10,292 pairs of the characters
- 220,579 conversations
- 304,713 utterances
- URL: http://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

Conversation Corpus

Characteristic of Conversations

- Length of utterances are various
- Length of conversations are various

Conversation Corpus

Movie script example)

DON CORLEONE:

You look terrible. I want you to eat well, to rest. And spend time with your family. And then, at the end of the month, this big shot will give you the part you want.

Utterance 1 (35 words)

JOHNNY:

It's too late. All the contracts have been signed, they're almost ready to shoot.

Utterance 2 (14 words)

DON CORLEONE:

I'll make him an offer he can't refuse.

Utterance 3 (8 words)

Conversation Corpus

Conversation Corpus

- Characteristic of Conversations
  - Length of utterances are various
  - Length of conversations are various
- Problem)
  How to give the conversations to a conversation model?
    - Train
    - Test
    - Inference

Conversation Corpus

Tensor format

	Conversation 1														
Utter 1	Love	the	new	album	every	single	song	is	incredible	congrats	girl	girl	gone	wild	by	Madonna
Utter 2	Please	come	on	stage	and	kiss	me	again	I	miss	you					
Utter 3	Tempting															
Utter 4	Are	you	gonna	make	me	work	for	this								
Utter 5	Why	of	course													
	Conversation 2														
Utter 1	Thanks	to	make	URL	It's	really	helpful	for	me							
Utter 2	glad	you	find	it	useful											

?

Conversation Corpus

Before padding

	Conversation 1															
Utter 1	Love	the	new	album	every	single	song	is	incredible	congrats	girl	girl	gone	wild	by	Madonna
Utter 2	Please	come	on	stage	and	kiss	me	again	I	miss	you					
Utter 3	Tempting															
Utter 4	Are	you	gonna	make	me	work	for	this								
Utter 5	Why	of	course													
	Conversation 2															
Utter 1	Thanks	to	make	URL	It's	really	helpful	for	me							
Utter 2	glad	you	find	it	useful											

After padding + length information

	Conversation 1															
Utter 1	Love	the	new	album	every	single	song	is	incredible	congrats	girl	girl	gone	wild	by	Madon a	*EOS*
Utter 2	Please		n	stage	and	kiss	me	again	I	miss	you	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	
Utter 3	Tempting	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*		*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 4	Are		gonna		me	work	for	this	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 5	Why	of	course	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*		*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
	Conversation 2																
Utter 1	Thanks	to	make	URL	It's	really	helpful	for	me	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 2	glad	you	find	it	useful	*EOS*	*PAD*	*PAD*	*PAD*		*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*

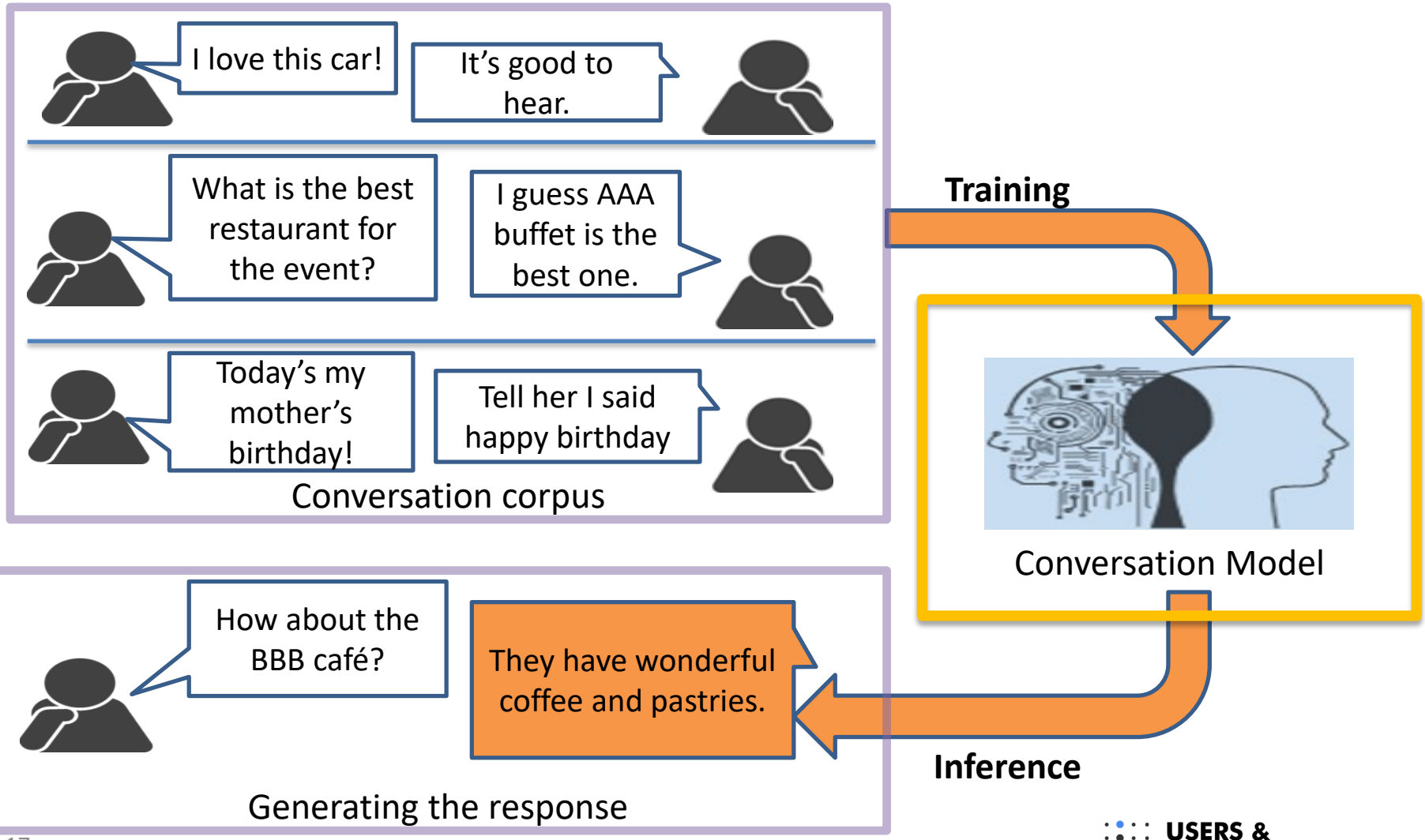
Code

```
def pad_utters_users(conversations, max_utters_length, max_conversation_length):
    def pad_tokens(tokens):
        n_valid_tokens = len(tokens)
        if n_valid_tokens > max_utters_length - 1:
            tokens = tokens[:max_utters_length - 1]
        n_pad = max_utters_length - n_valid_tokens - 1
        tokens = tokens + [st.EOS_TOKEN] + [st.PAD_TOKEN] * n_pad
        return tokens

    def pad_conversation(one_conversation):
        return [pad_tokens(utter) for utter in one_conversation]

    all_padded_utters = list()
    all_utter_length = list()

    for conversation in conversations:
        if len(conversation) > max_conversation_length:
            conversation = conversation[:max_conversation_length]
        utter_length = [min(len(utter) + 1, max_utters_length) for utter in conversation]
        all_utter_length.append(utter_length)

        utters = pad_conversation(conversation)
        all_padded_utters.append(utters)

    return all_padded_utters, all_utter_length
```

Twitter_Conversation_Preprocess.py

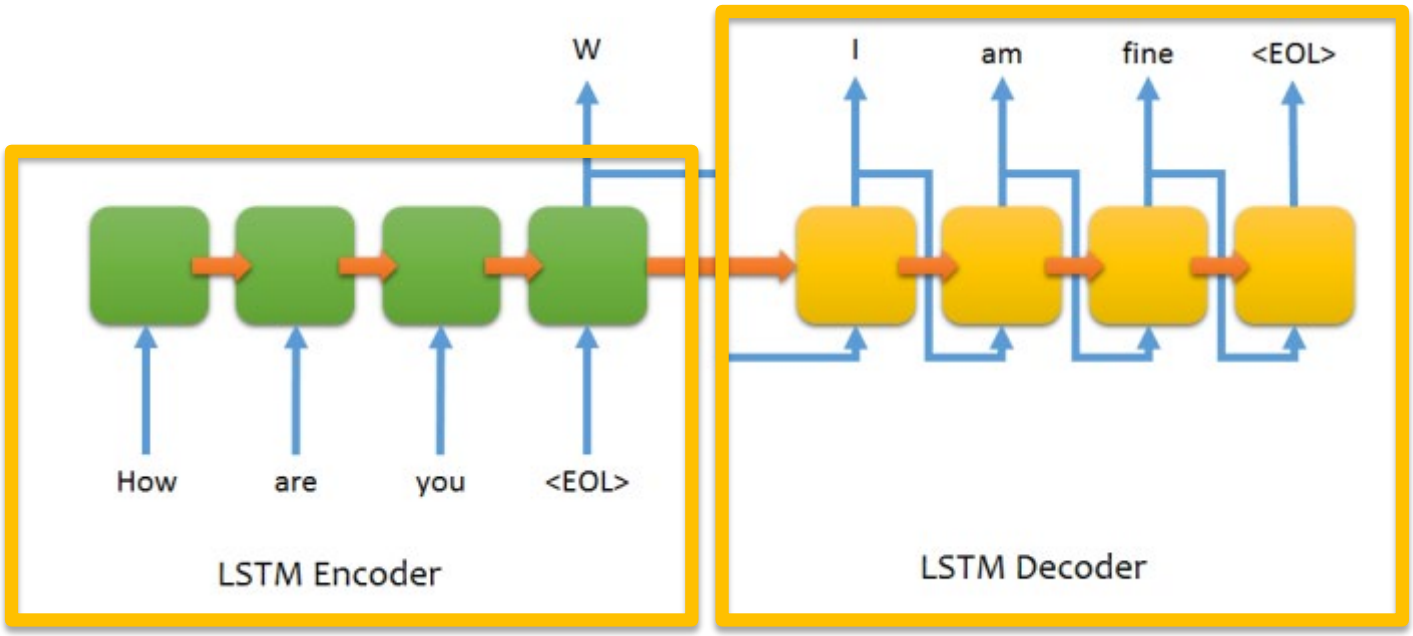
HRED

Conversation Model



USERS & INFORMATION

Seq2Seq



Conversation Corpus

Movie script example)

DON CORLEONE:

You look terrible. I want you to eat well, to rest. And spend time with your family. And then, at the end of the month, this big shot will give you the part you want.

Utterance 1 (35 words)

JOHNNY:

It's too late. All the contracts have been signed, they're almost ready to shoot.

Utterance 2 (14 words)

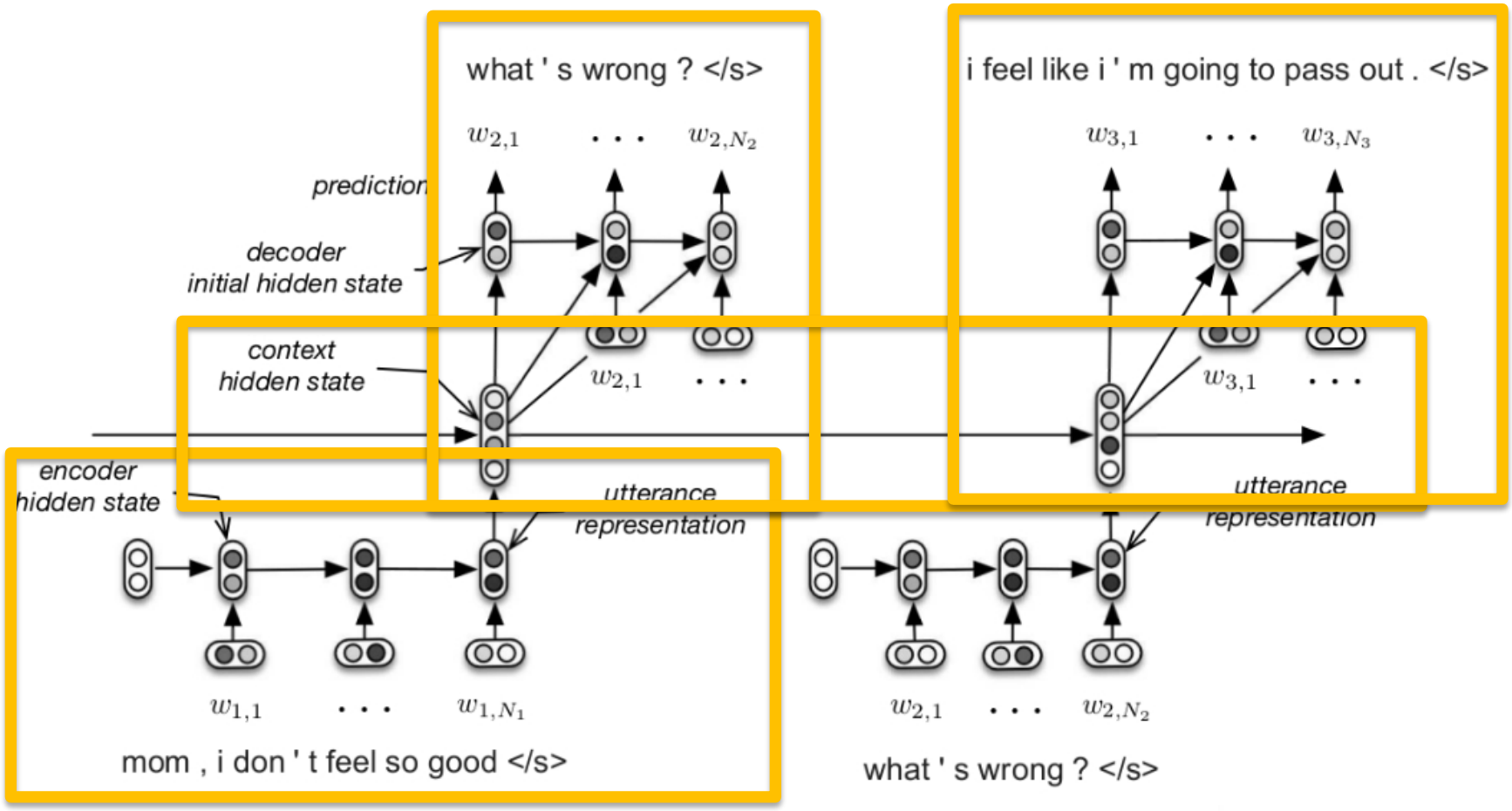
DON CORLEONE:

I'll make him an offer he can't refuse.

Utterance 3 (8 words)

HRED

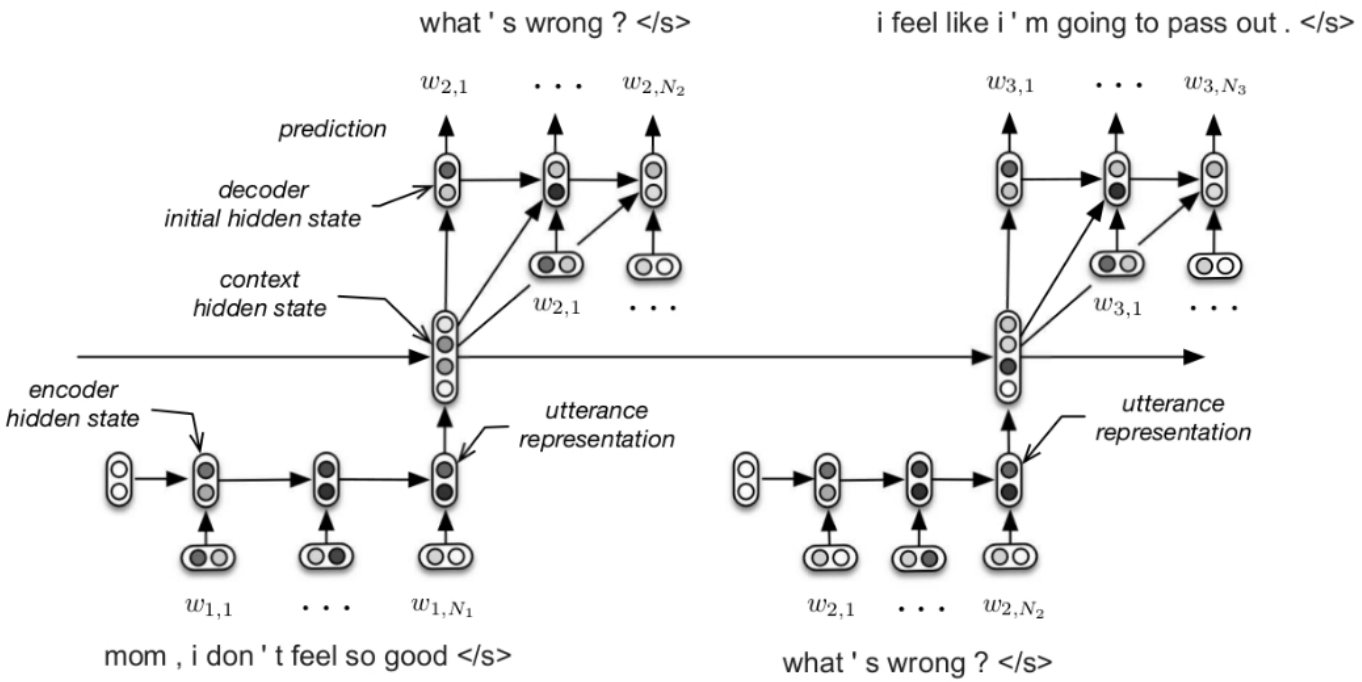
Hierarchical Recurrent Encoder-Decoder



USERS & INFORMATION

HRED

- Encoder RNN
- Context RNN
- Decoder RNN

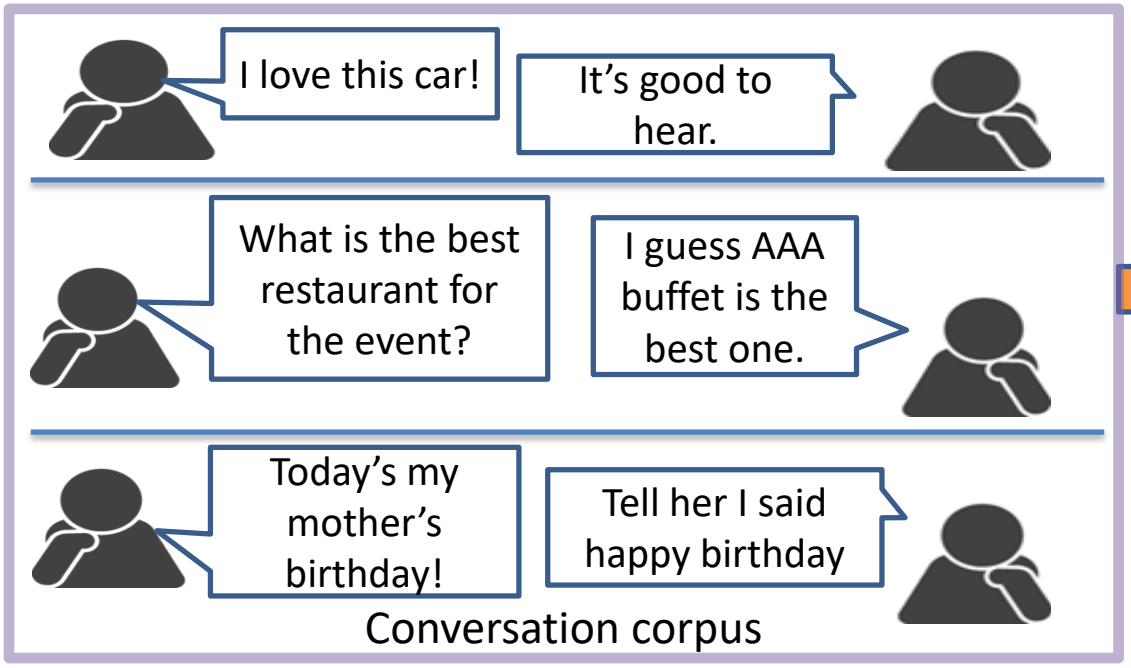


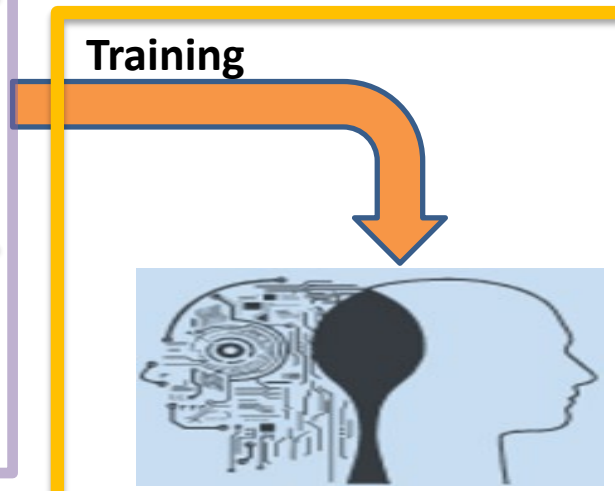
HRED - Code

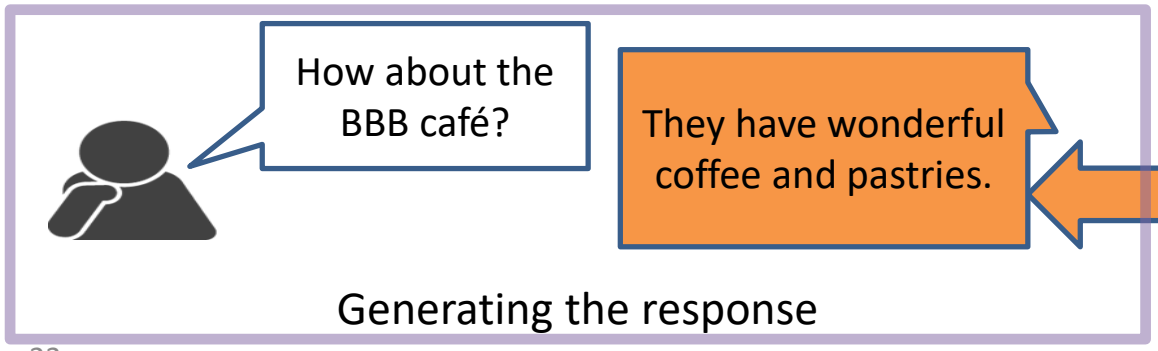
```
class HRED(nn.Module):
    def __init__(self, config):
        super(HRED, self).__init__()

        self.config = config
        self.encoder = layers.EncoderRNN(config.vocab_size, config.embedding_size, config.encoder_hidden_size,
                                         config.rnn, config.num_layers, config.bidirectional, config.dropout,
                                         pretrained_wv_path=config.pretrained_wv_path)

        context_input_size = (config.num_layers * config.encoder_hidden_size * self.encoder.num_directions)
        self.context_encoder = layers.ContextRNN(context_input_size, config.context_size, config.rnn,
                                                 config.num_layers, config.dropout)

        self.decoder = layers.DecoderRNN(config.vocab_size, config.embedding_size, config.decoder_hidden_size,
                                         config.rnncell, config.num_layers, config.dropout, config.word_drop,
                                         config.max_unroll, config.sample, config.temperature, config.beam_size)

        self.context2decoder = layers.FeedForward(config.context_size,
                                                  config.num_layers * config.decoder_hidden_size,
                                                  num_layers=1, activation=config.activation)

        if config.tie_embedding:
            self.decoder.embedding = self.encoder.embedding
```

models/hred.py

Conversation Model







Inference

Train

Environment

- SW
  - Python 3.7.3
  - PyTorch 1.1.0
- HW
  - Intel Xeon CPU E5-2640
  - 256GB
  - GTX 1080TI

Train

Operation

bash RunTrain.sh 0 cornell HRED 30 50

- 0: GPU ID
- HRED: model name
- 30: batch size
- 50: maximum epoch

```
export CUDA_VISIBLE_DEVICES=$1

python train.py --data="$2" --model="$3"
--batch_size="$4" --eval_batch_size="$4" --n_epoch="$5"
```

Train - Code

```
def train(self):
    epoch_loss_history = list()
    min_validation_loss = sys.float_info.max
    patience_cnt = self.config.patience

    for epoch_i in range(self.epoch_i, self.config.n_epoch):
        batch_loss_history = list()
        self.model.train()
        for batch_i, (conversations, convs_length, utterances_length) in \
                #   A conversation: [max_conv_len, max_utter_len] list of utterances
                #   An utterance: [max_utter_len] list of tokens
                # convs_length: [batch_size] list of integer
                # utterances_length: [batch_size, max_conv_len] list of conversation that has a list of utterance length
```

solvers/hred_solver.py

Train - Code

```
input_conversations = [conv[:-1] for conv in conversations]
target_conversations = [conv[1:] for conv in conversations]

input_utterances = [utter for conv in input_conversations for utter in conv]
target_utterances = [utter for conv in target_conversations for utter in conv]
input_utterance_length = [l for len_list in utterances_length for l in len_list[:-1]]
target_utterance_length = [l for len_list in utterances_length for l in len_list[1:]]
input_conversation_length = [conv_len - 1 for conv_len in convs_length]

input_utterances = to_var(torch.LongTensor(input_utterances))
target_utterances = to_var(torch.LongTensor(target_utterances))
input_utterance_length = to_var(torch.LongTensor(input_utterance_length))
target_utterance_length = to_var(torch.LongTensor(target_utterance_length))
input_conversation_length = to_var(torch.LongTensor(input_conversation_length))

self.optimizer.zero_grad()

utterances_logits = self.model(input_utterances, input_utterance_length,
                               input_conversation_length, target_utterances, decode=False)

batch_loss, n_words = masked_cross_entropy(utterances_logits, target_utterances, target_utterance_length)
```

solvers/hred_solver.py

Train - Code

```
def forward(self, input_utterances, input_utterance_length, input_conversation_length,
            target_utterances, decode=False):
    """
    Forward of HRED
    :param input_utterances: [num_utterances, max_utter_len]
    :param input_utterance_length: [num_utterances]
    :param input_conversation_length: [batch_size]
    :param target_utterances: [num_utterances, seq_len]
    :param decode: True or False
    :return: decoder_outputs
    """
```

models/hred.py

Train - Code

```
num_utterances = input_utterances.size(0)
max_conv_len = input_conversation_length.data.max().item()

encoder_outputs, encoder_hidden = self.encoder(input_utterances, input_utterance_length)
encoder_hidden = encoder_hidden.transpose(1, 0).contiguous().view(num_utterances, -1)
start = torch.cumsum(torch.cat((to_var(input_conversation_length.data.new(1).zero_()),
                                input_conversation_length[:-1])), 0)

encoder_hidden = torch.stack([pad(encoder_hidden.narrow(0, s, l), max_conv_len)
                              for s, l in zip(start.data.tolist(),
                                              input_conversation_length.data.tolist())], 0)
```

	Conversation 1																
Utter 1	Love	the	new	album	every	single	song	is	incredible	congrats	girl	girl	gone	wild	by	Madon	a *EOS*
Utter 2	Please	come	on	stage	and	kiss	me	again	I	miss	you	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 3	Tempting	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 4	Are	you	gonna	make	me	work	for	this	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 5	Why	of	course	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
	Conversation 2																
Utter 1	Thanks	to	make	URL	It's	really	helpful	for	me	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*
Utter 2	glad	you	find	it	useful	*EOS*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*	*PAD*

```
else:
    prediction, final_score, length = self.decoder.beam_decode(init_h=decoder_init)
    return prediction
```

models/hred.py

Train - Code

```
num_utterances = input_utterances.size(0)
max_conv_len = input_conversation_length.data.max().item()

encoder_outputs, encoder_hidden = self.encoder(input_utterances, input_utterance_length)
encoder_hidden = encoder_hidden.transpose(1, 0).contiguous().view(num_utterances, -1)
start = torch.cumsum(torch.cat((to_var(input_conversation_length.data.new(1).zero_()),
                                input_conversation_length[:-1])), 0)

encoder_hidden = torch.stack([pad(encoder_hidden.narrow(0, s, l), max_conv_len)
                              for s, l in zip(start.data.tolist(),
                                              input_conversation_length.data.tolist())], 0)

context_outputs, context_last_hidden = self.context_encoder(encoder_hidden, input_conversation_length)
context_outputs = torch.cat([context_outputs[i, l-1, :]
                             for i, l in enumerate(input_conversation_length.data)])

decoder_init = self.context2decoder(context_outputs)
decoder_init = decoder_init.view(self.decoder.num_layers, -1, self.decoder.hidden_size)

if not decode:
    decoder_outputs = self.decoder(target_utterances, init_h=decoder_init, decode=decode)
    return decoder_outputs

else:
    prediction, final_score, length = self.decoder.beam_decode(init_h=decoder_init)
    return prediction
```

models/hred.py

Train - Code

```
batch_loss, n_words = masked_cross_entropy(utterances_logits, target_utterances, target_utterance_length)

assert not isnan(batch_loss.item())
batch_loss_history.append(batch_loss.item())
n_total_words += n_words.item()

if batch_i % self.config.print_every == 0:
    tqdm.write(f'Epoch: {epoch_i+1}, iter {batch_i}: loss = {batch_loss.item()/ n_words.item():.3f}')

batch_loss.backward()
torch.nn.utils.clip_grad_norm_(self.model.parameters(), self.config.clip)
self.optimizer.step()
```

solvers/hred_solver.py

Train - Code

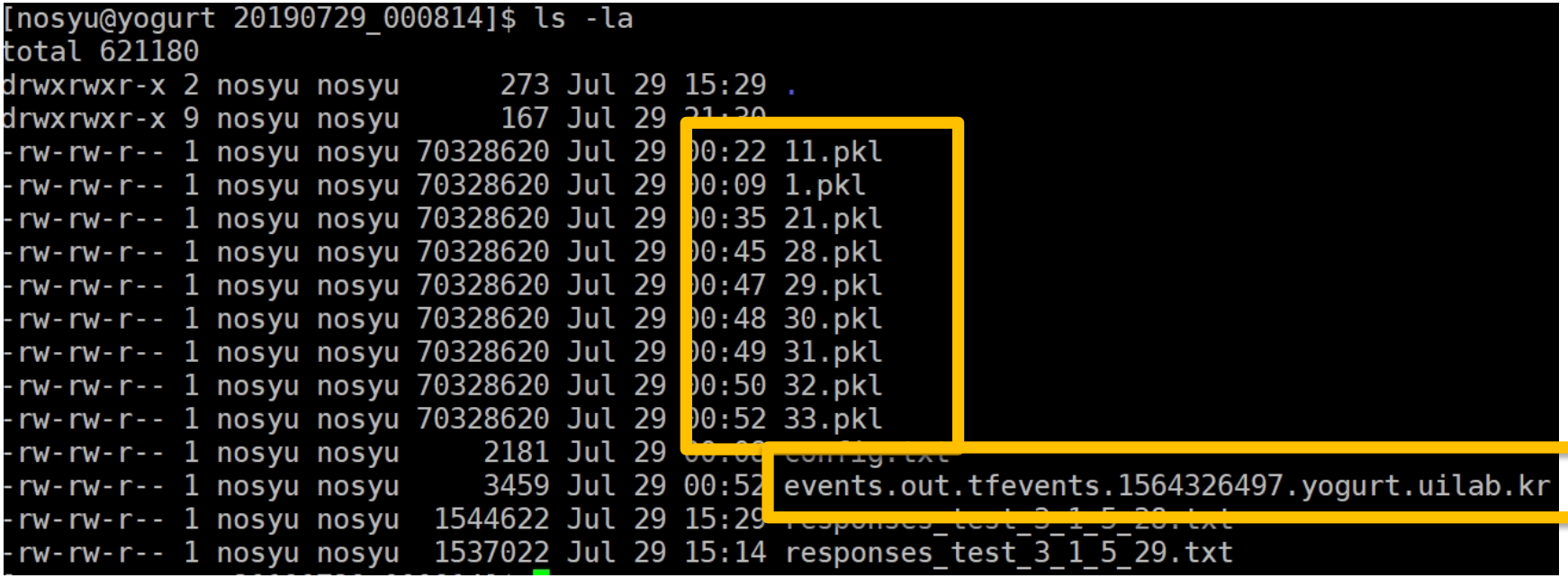
```
print('\n<Validation>...')
self.validation_loss = self.evaluate()

if epoch_i % self.config.plot_every_epoch == 0:
    self.write_summary(epoch_i)

if min_validation_loss > self.validation_loss:
    min_validation_loss = self.validation_loss
else:
    patience_cnt -= 1
    self.save_model(epoch_i)

if patience_cnt < 0:
    print(f'\nEarly stop at {epoch_i}')
    self.save_model(epoch_i)
    return epoch_loss_history
```

solvers/hred_solver.py

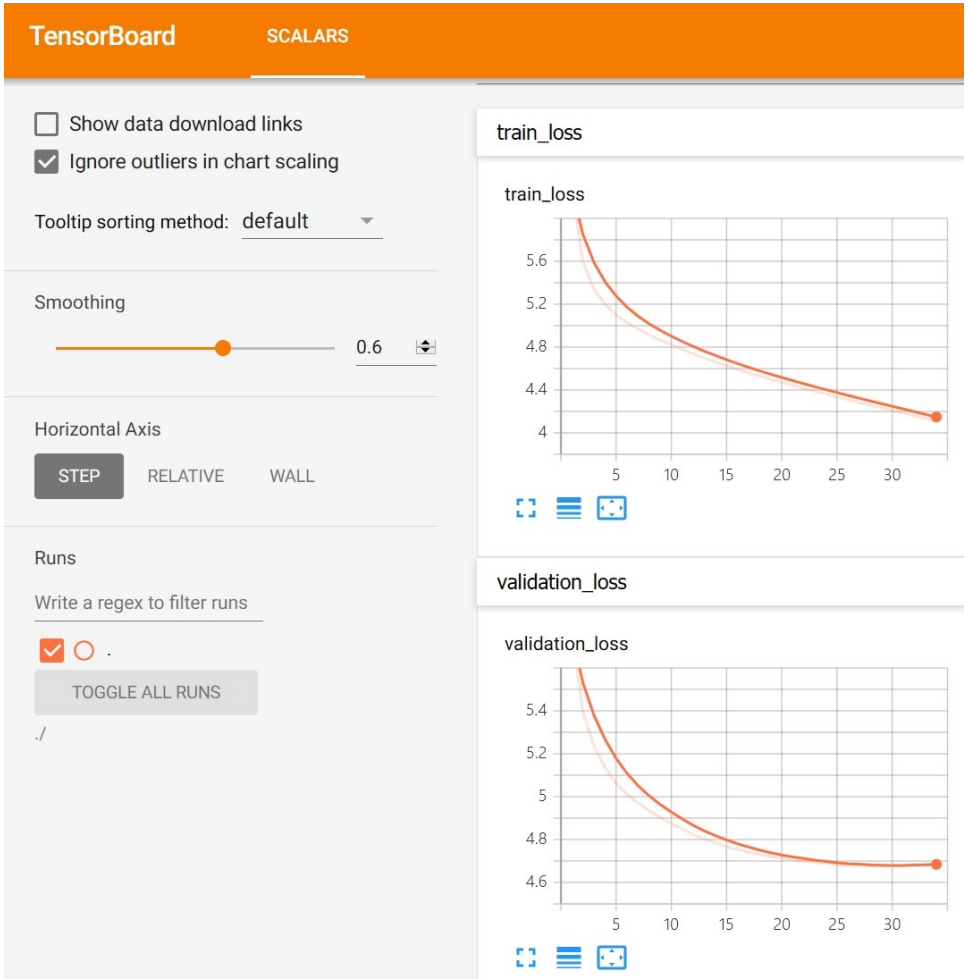
Train - Code

```
[nosyu@yogurt src]$ bash Run_train_TC.sh 0 HRED 30 50 True False
Vocabulary size: 14518
Load the wv Done
/home/nosyu/anaconda3/lib/python3.7/site-packages/torch/nn/modules/rnn.py:46: UserWarning: dropout
, so non-zero dropout expects num_layers greater than 1, but got dropout=0.2 and num_layers=1
  "num_layers={}".format(dropout, num_layers))
Parameter initiailization
        encoder.rnn.weight_hh_l0
        encoder.rnn.bias_hh_l0
        encoder.rnn.weight_hh_l0_reverse
        encoder.rnn.bias_hh_l0_reverse
        context_encoder.rnn.weight_hh_l0
        context_encoder.rnn.bias_hh_l0
        decoder.rnncell.layers.0.weight_hh
        decoder.rnncell.layers.0.bias_hh
Epoch: 1, iter 0: loss = 9.569
Epoch: 1, iter 100: loss = 6.308
Epoch: 1, iter 200: loss = 6.280
Epoch: 1, iter 300: loss = 6.168
Epoch: 1, iter 400: loss = 5.967
Epoch: 1, iter 500: loss = 5.841
Epoch: 1, iter 600: loss = 5.867
100%|                                                  | 659/659 [01:17<00:00,  8.79it/s]
Epoch 1 loss average: 6.272
Save parameters to /data/nosyu/git/conversation-models/results/tc_10_10/HRED/2019_0729_213029/1.pkl

<Validation>...
100%|                                                   | 81/81 [00:02<00:00, 31.04it/s]
Validation loss: 5.747

Epoch: 2, iter 0: loss = 5.785
 11%|    □                                             | 70/659 [00:08<01:08,  8.62it/s]
```

Train - Result

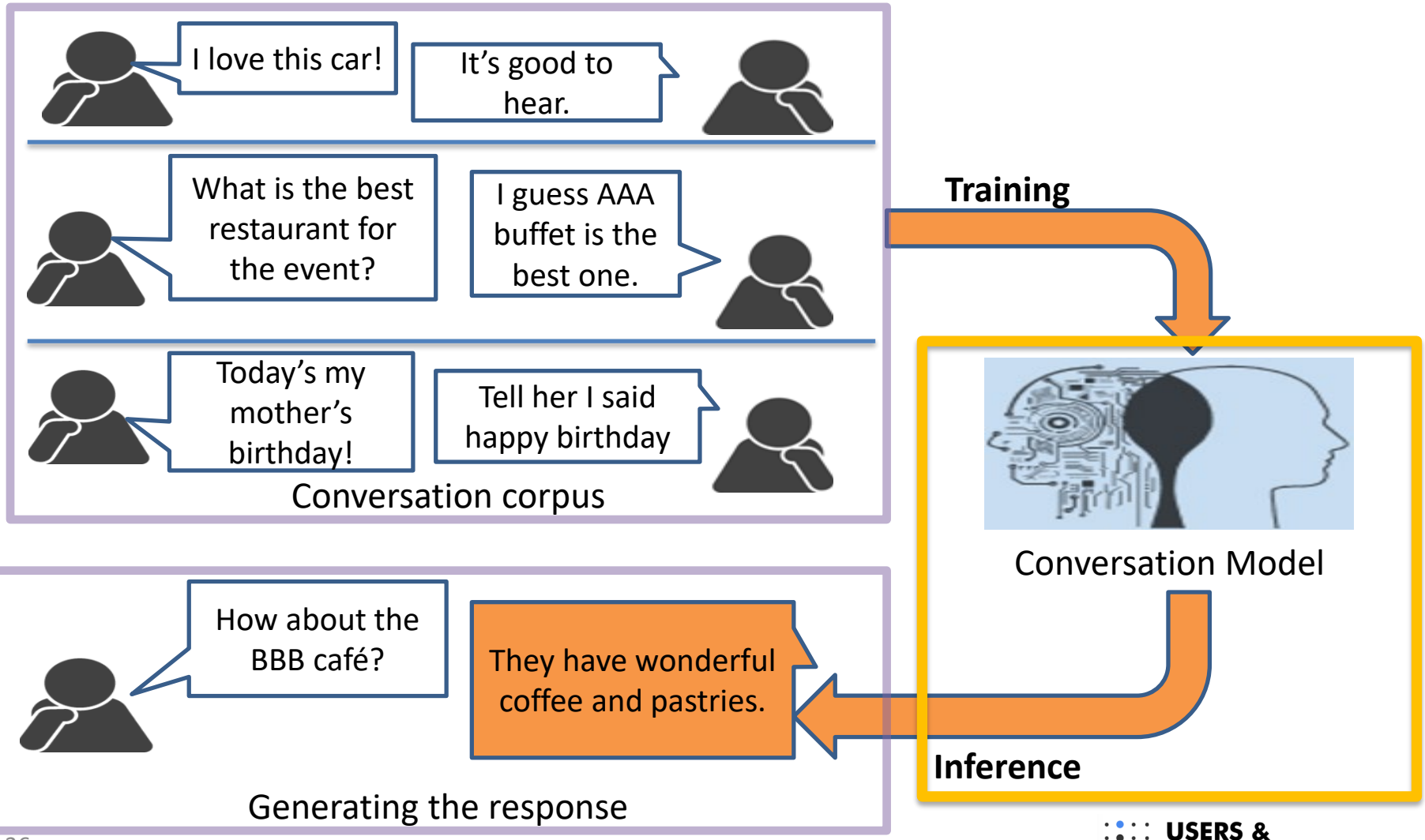


Train - Result



USERS & INFORMATION

Conversation Model



Inference

Operation

bash RunExportTestSamples.sh 0 HRED 30 2 1 30.pkl

- 0: GPU ID
- HRED: model name
- 30: batch size
- 2: number of context utterance
- 1: number of sample step
- 30.pkl: saved model file

Conversation Corpus

Ex) Context: 2, Sample step: 1

DON CORLEONE:

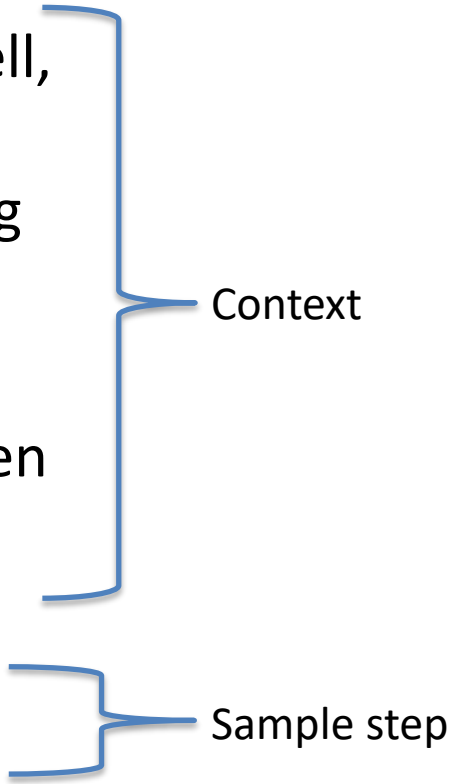
You look terrible. I want you to eat well, to rest. And spend time with your family. And then, at the end of the month, this big shot will give you the part you want.

JOHNNY:

It's too late. All the contracts have been signed, they're almost ready to shoot.

DON CORLEONE:

I'll make him an offer he can't refuse.



Inference - Code

```
def generate(self, context, utterances_length, n_context):
    """
    Generate the response based on the context
    :param context: [batch_size, n_context, max_utter_len] given conversation utterances
    :param utterances_length: [batch_size, n_context] length of the utterances in the context
    :param n_context: length of the context turns
    :return: generated responses
    """
```

models/hred.py

Inference - Code

```
for i in range(n_context):
    encoder_outputs, encoder_hidden = self.encoder(context[:, i, :], utterances_length[:, i])

    encoder_hidden = encoder_hidden.transpose(1, 0).contiguous().view(batch_size, -1)
    context_outputs, context_hidden = self.context_encoder.step(encoder_hidden, context_hidden)
```

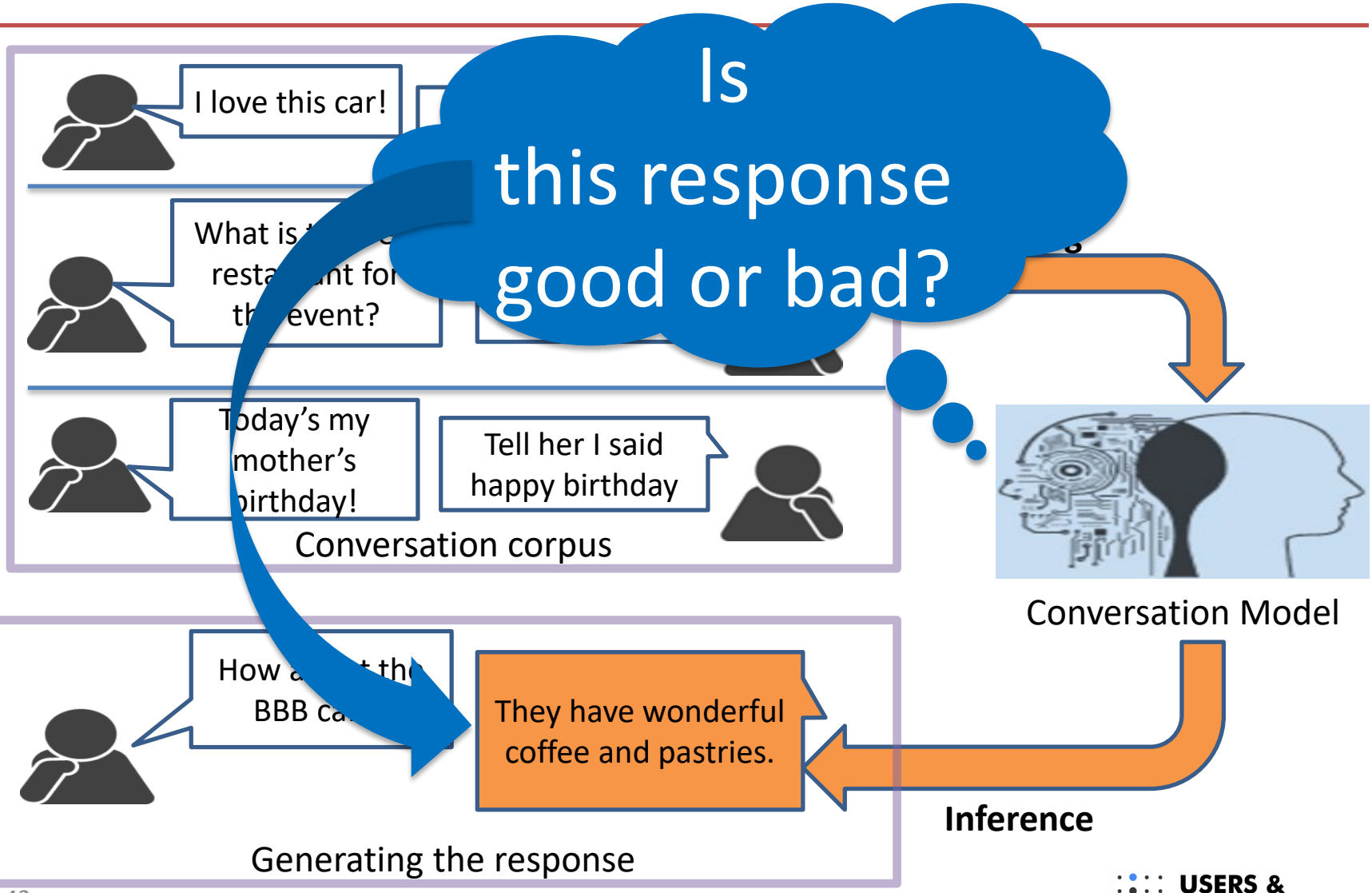
```
for j in range(self.config.n_sample_step):
    context_outputs = context_outputs.squeeze(1)
    decoder_init = self.context2decoder(context_outputs)
    decoder_init = decoder_init.view(self.decoder.num_layers, -1, self.decoder.hidden_size)

    prediction_all, final_score, length = self.decoder.beam_decode(init_h=decoder_init)
    all_samples.append(prediction_all)
    prediction = prediction_all[:, 0, :]
    length = [l[0] for l in length]
    length = to_var(torch.LongTensor(length))
    samples.append(prediction)

    encoder_outputs, encoder_hidden = self.encoder(prediction, length)
    encoder_hidden = encoder_hidden.transpose(1, 0).contiguous().view(batch_size, -1)
    context_outputs, context_hidden = self.context_encoder.step(encoder_hidden, context_hidden)
```

models/hred.py

Inference - Result

```
[nosyu@yogurt 20190729_000814]$ ls -la
total 621180
drwxrwxr-x 2 nosyu nosyu      273 Jul 29 15:29 .
drwxrwxr-x 9 nosyu nosyu      167 Jul 29 21:30 ..
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:22 11.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:09 1.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:35 21.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:45 28.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:47 29.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:48 30.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:49 31.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:50 32.pkl
-rw-rw-r-- 1 nosyu nosyu 70328620 Jul 29 00:52 33.pkl
-rw-rw-r-- 1 nosyu nosyu     2181 Jul 29 00:08 config.txt
-rw-rw-r-- 1 nosyu nosyu     3459 Jul 29 00:52 events.out.tfevents.1564320407.yogurt.uilab.kr
-rw-rw-r-- 1 nosyu nosyu  1544622 Jul 29 15:29 responses_test_3_1_5_28.txt
-rw-rw-r-- 1 nosyu nosyu  1537022 Jul 29 15:14 responses_test_3_1_5_29.txt
```

Conversation Model



USERS & INFORMATION

Evaluation

- Length
  - Count the number of words in generated response
- BLEU
  - Compare generated response with ground-truth
  - Count the overlap of n-gram
  - https://www.aclweb.org/anthology/P02-1040/
- ROUGE-L
  - Compare generated response with ground-truth
  - Count the overlap of n-gram in a longest common subsequence
  - https://www.aclweb.org/anthology/W04-1013/

Evaluation

Operation

bash RunEval.sh responses_test_2_1_5_30.txt

– responses_test_2_1_5_20.txt: generated responses file

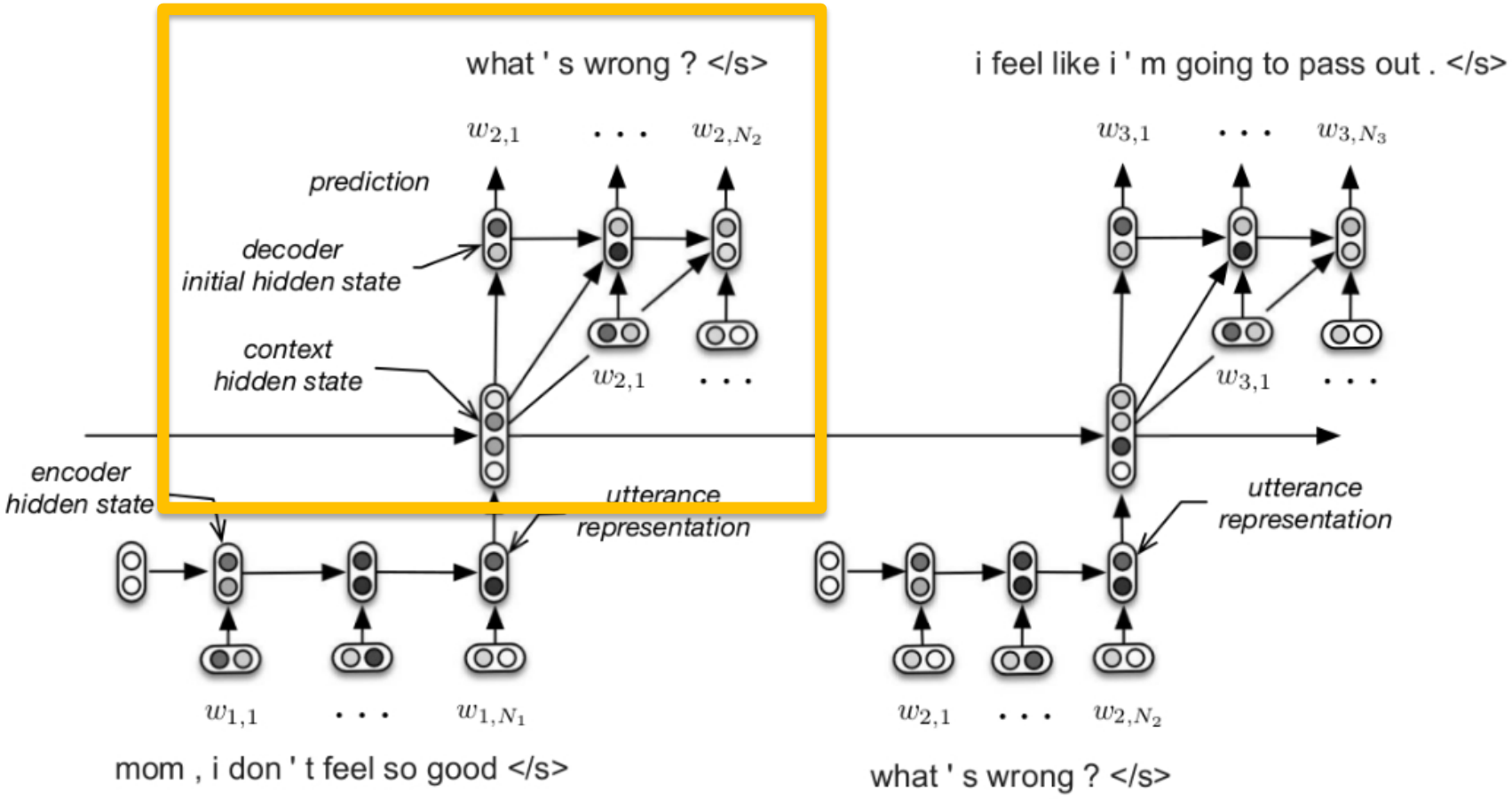
Evaluation - Result

```
[nosyu@yogurt src]$ bash RunEval.sh ../results/c
Metric                  Average    Standard Error
----------------------  ---------  ----------------
Length                  10.5855           0.0891897
BLEU                     0.195279         0.00276903
ROUGE-L Precision        0.0982376        0.00273272
ROUGE-L Recall           0.145718         0.00348285
ROUGE-L F1               0.0845265        0.00209298
```

VARIATIONAL HIERARCHICAL RECURRENT ENCODER DECODER

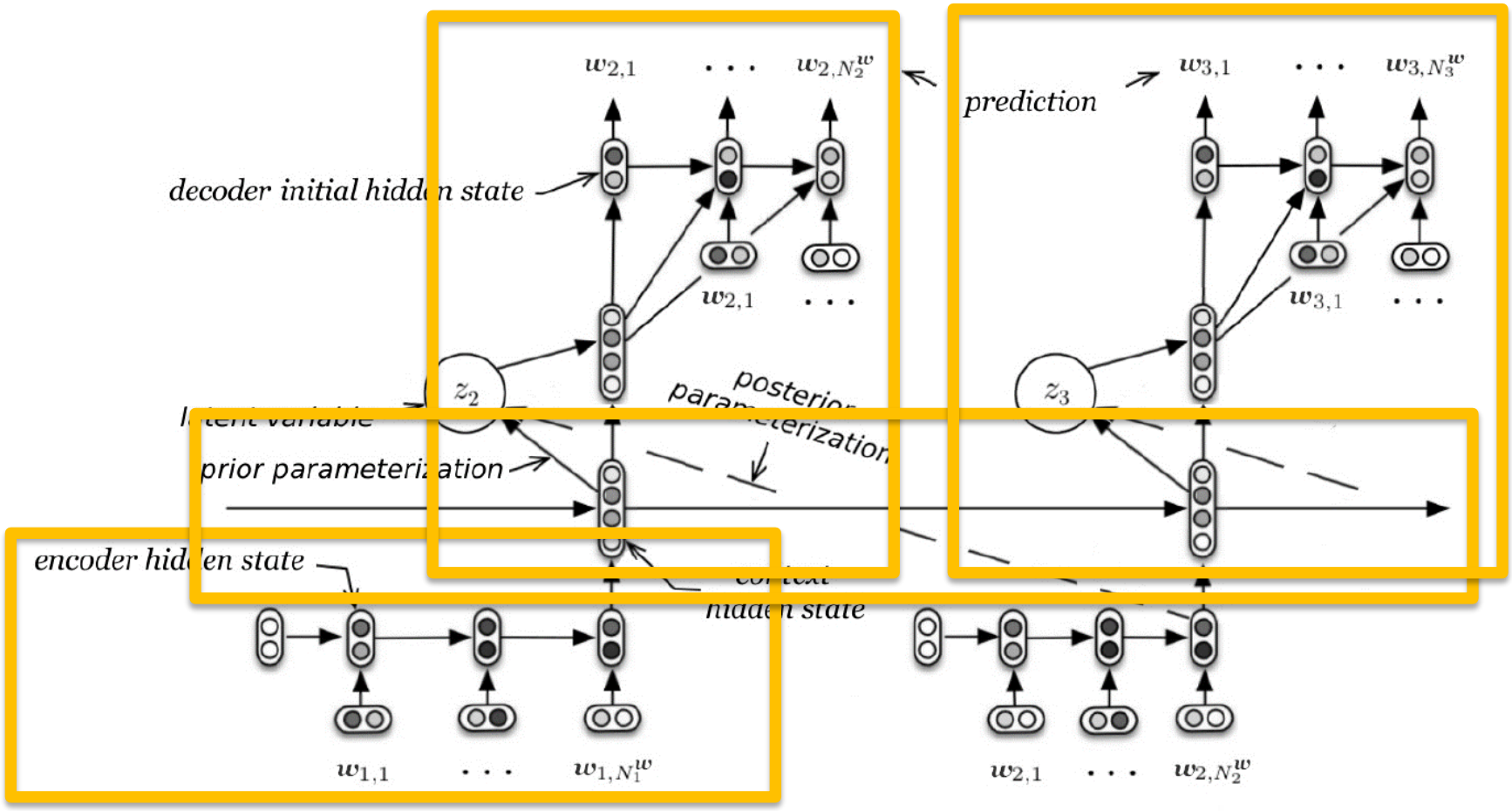
HRED

Hierarchical Recurrent Encoder-Decoder



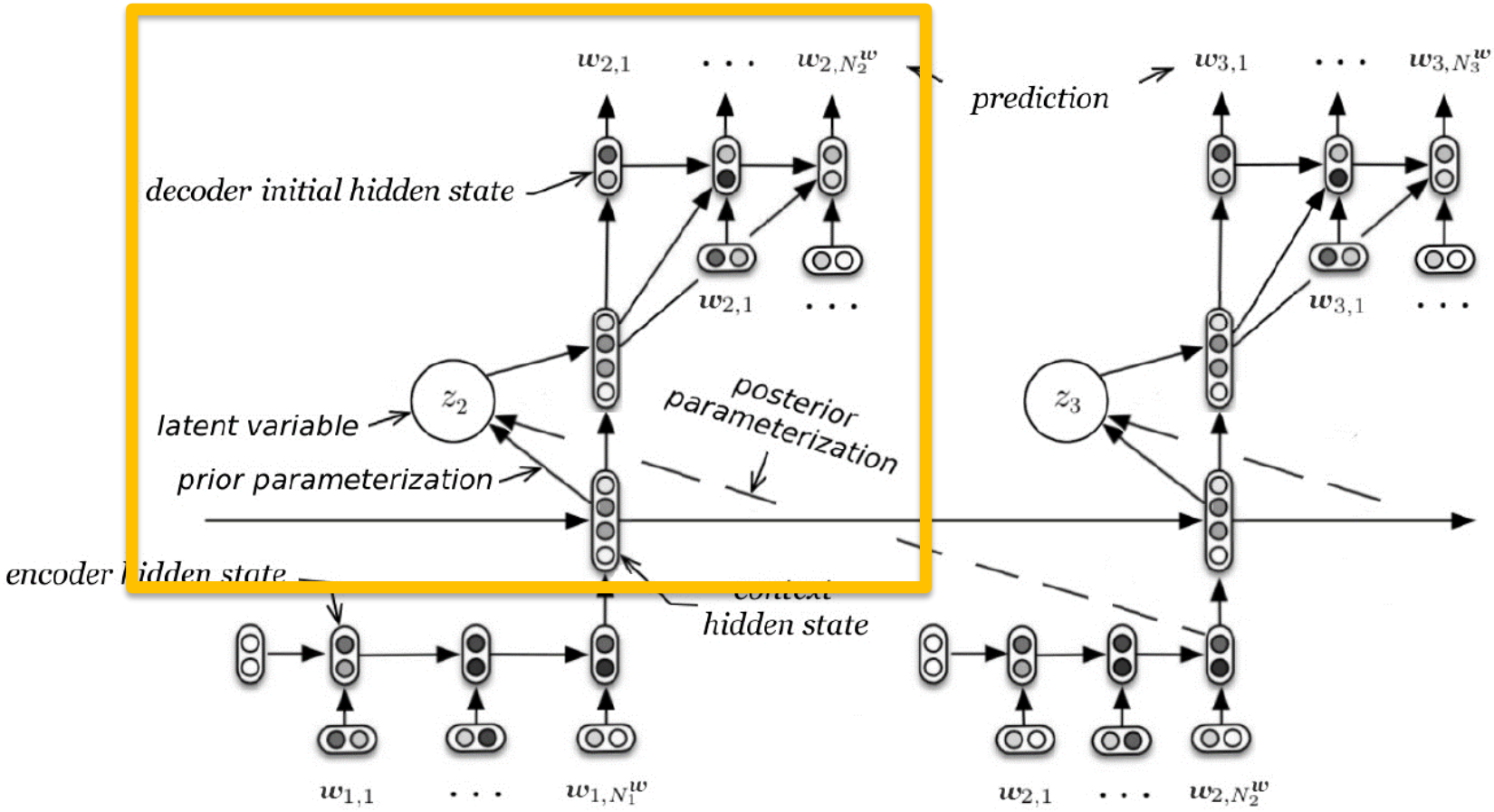
VHRED

Variational Hierarchical Recurrent Encoder Decoder

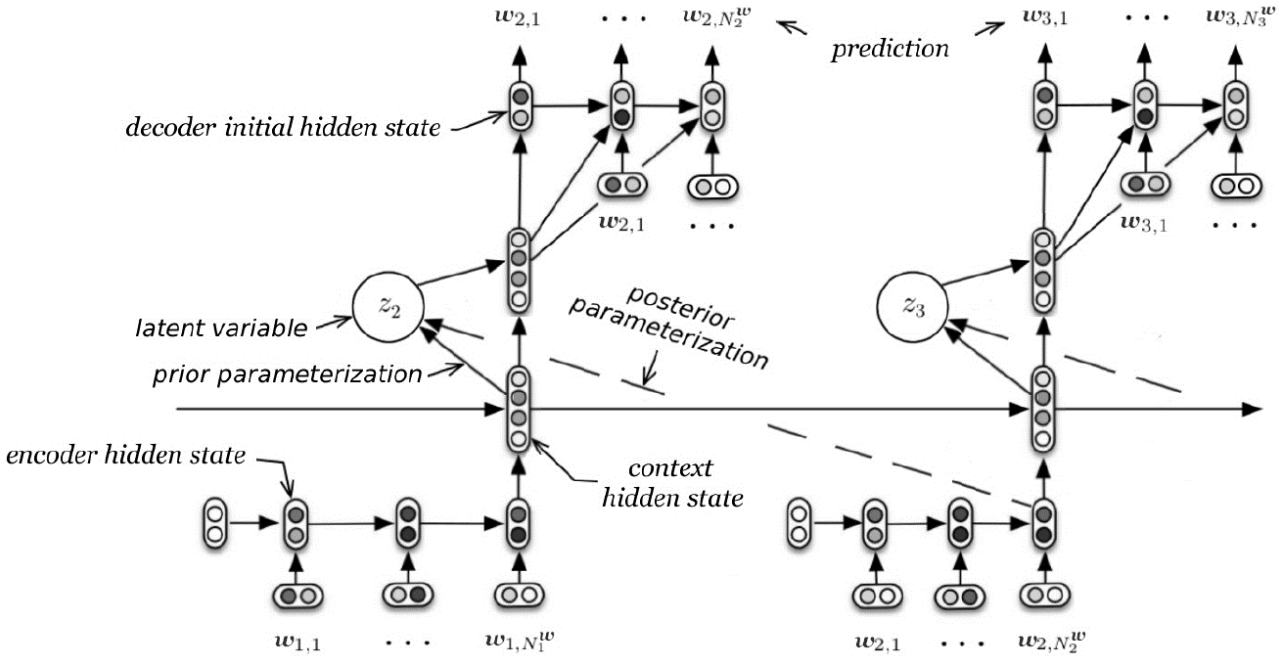


VHRED

Variational Hierarchical Recurrent Encoder Decoder



VHRED



$$
\begin{aligned}
\mathbf{h}^{\text{enc}}_{t-1} &= f^{\text{enc}}_{\boldsymbol{\theta}}(\mathbf{x}_{t-1}) \\
\mathbf{h}^{\text{cxt}}_{t} &= f^{\text{cxt}}_{\boldsymbol{\theta}}(\mathbf{h}^{\text{cxt}}_{t-1}, \mathbf{h}^{\text{enc}}_{t-1}) \\
p_{\boldsymbol{\theta}}(\mathbf{z}^{\text{utt}}_{t} | \mathbf{x}_{<t}) &= \mathcal{N}(\mathbf{z} | \boldsymbol{\mu}_t, \boldsymbol{\sigma}_t \mathbf{I}) \\
\text{where } \boldsymbol{\mu}_t &= \text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}^{\text{cxt}}_{t}) \\
\boldsymbol{\sigma}_t &= \text{Softplus}(\text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}^{\text{cxt}}_{t})) \\
p_{\boldsymbol{\theta}}(\mathbf{x}_t | \mathbf{x}_{<t}) &= f^{\text{dec}}_{\boldsymbol{\theta}}(\mathbf{x} | \mathbf{h}^{\text{cxt}}_{t}, \mathbf{z}^{\text{utt}}_{t})
\end{aligned}
$$

VHRED

- Prior

$$p_{\boldsymbol{\theta}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{<t}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}_t, \boldsymbol{\sigma}_t\mathbf{I})$$
$$\text{where } \boldsymbol{\mu}_t = \text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_t^{\text{cxt}})$$
$$\boldsymbol{\sigma}_t = \text{Softplus}(\text{MLP}_{\boldsymbol{\theta}}(\mathbf{h}_t^{\text{cxt}}))$$

- Variational posterior

$$q_{\boldsymbol{\phi}}(\mathbf{z}_t^{\text{utt}}|\mathbf{x}_{\leq t}) = \mathcal{N}(\mathbf{z}|\boldsymbol{\mu}'_t, \boldsymbol{\sigma}'_t I)$$
$$\text{where } \boldsymbol{\mu}'_t = \text{MLP}_{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{h}_t^{\text{cxt}})$$
$$\boldsymbol{\sigma}'_t = \text{Softplus}(\text{MLP}_{\boldsymbol{\phi}}(\mathbf{x}_t, \mathbf{h}_t^{\text{cxt}}))$$

VHRED - Code

```
def forward(self, utterances, utterance_length, input_conversation_length, target_utterances,
            decode=False):
    """
    Forward of VHRED
    :param utterances: [num_utterances, max_utter_len]
    :param utterance_length: [num_utterances]
    :param input_conversation_length: [batch_size]
    :param target_utterances: [num_utterances, seq_len]
    :param decode: True or False
    :return: decoder_outputs
    """
```

models/vhred.py

Tasks

1. Complete the code of VHRED
   Fill two functions for variable z_t^{utt} in the *vhred.py*

```
def prior(self, context_outputs):
    """
    Compute prior
    :param context_outputs: h_t^{cxt} [num_true_utterances, context_rnn_output_size]
    :return: [mu, sigma]
    """
    pass
```

```
def posterior(self, context_outputs, encoder_hidden):
    """
    Compute variational posterior
    :param context_outputs: h_t^{cxt} [num_true_utterances, context_rnn_output_size]
    :param encoder_hidden: x_t [num_true_utterances, encoder_rnn_output_size]
    :return: [mu, sigma]
    """
    pass
```

Tasks

1. Complete the code of VHRED
   Fill two functions for variable z_t^{utt} in the *vhred.py*
2. Run HRED and VHRED codes with *cornell* corpus
   1. Train the model
   2. Generate the responses
   3. Evaluate the responses
3. Write about introduced evaluation metrics
   1. Limitations of the metrics
   2. Suggestions for evaluation

Thank you!

Any questions or comments?

JinYeong Bak

jy.bak@kaist.ac.kr

U&I Lab, KAIST