지능형 대화 에이전트 (Chatbot) 실습

1. Deep learning basic with PyTorch
2. Deep learning based Chatbot
3. ParlAI: a framework for dialog AI reserach
4. Practice

** Prerequisite : Python, Neural Network*

2017. 9. 8.

KAIST

CNSL Computational NeuroSystems Laboratory

Deep learning basic with PyTorch

References:

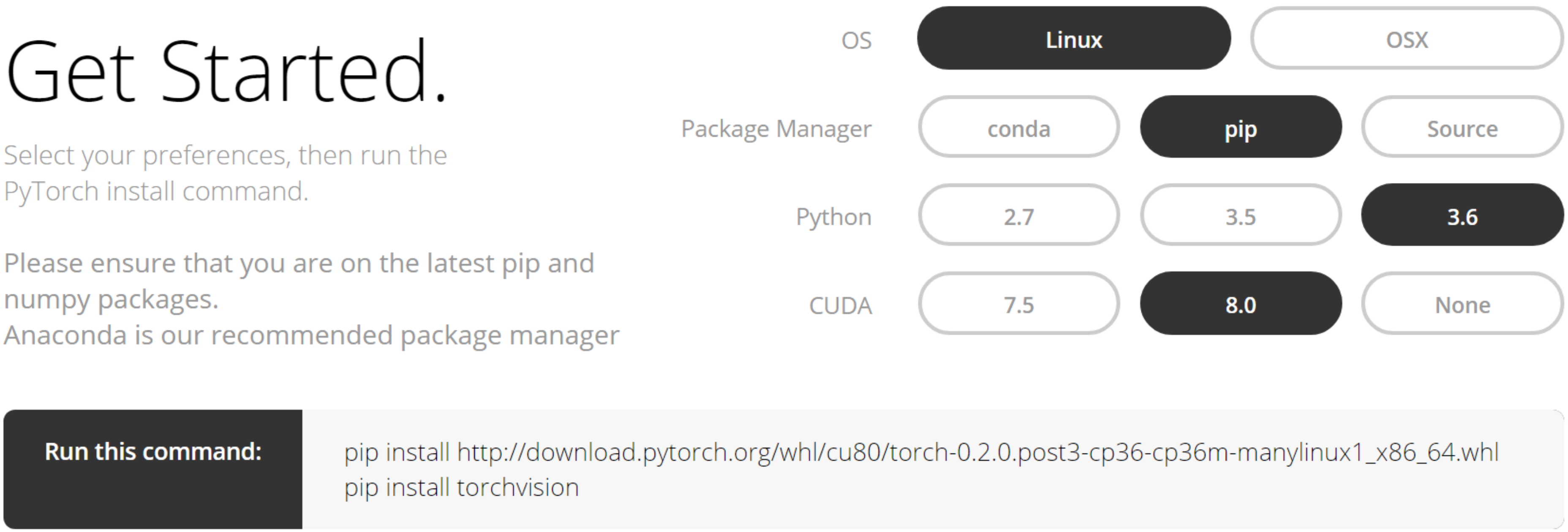
http://pytorch.org/tutorials/beginner/pytorch_with_examples.html

Pytorch

Python based scientific computing package targeted at two sets of audiences:

- A replacement for numpy to use the power of GPUs
- a deep learning research platform that provides maximum flexibility and speed

Installiation



- http://pytorch.org/
- we recommend you to prepare CUDA-enable computing environment.

Basic examples for Pytorch

- Tensors
- with Numpy
- GPU
- Autograd
- Neural Network
- Loss
- Optimizer

source :
[1] http://pytorch.org/tutorials/beginner/deep_learning_60min_blitz.html
[2] http://pytorch.org/tutorials/beginner/pytorch_with_examples.html

PLEASE FOLLOW ALL EXAMPLES IN ABOVE LINK BY YOURSELF

Tensors

```
from __future__ import print_function
import torch
```

Construct a 5x3 matrix, uninitialized:

```
x = torch.Tensor(5, 3)
print(x)
```

Out:

```
-2.9226e-26  1.5549e-41  1.5885e+14
 0.0000e+00  7.0065e-45  0.0000e+00
 7.0065e-45  0.0000e+00  4.4842e-44
 0.0000e+00  4.6243e-44  0.0000e+00
 1.5810e+14  0.0000e+00  1.6196e+14
[torch.FloatTensor of size 5x3]
```

Converting torch Tensor to numpy Array

```
a = torch.ones(5)
print(a)
```

Out:

```
1
 1
 1
 1
 1
[torch.FloatTensor of size 5]
```

```
b = a.numpy()
print(b)
```

Out:

```
[ 1.  1.  1.  1.  1.]
```

Converting numpy Array to torch Tensor

See how changing the np array changed the torch Tensor automatically

```
import numpy as np
a = np.ones(5)
b = torch.from_numpy(a)
np.add(a, 1, out=a)
print(a)
print(b)
```

Out:

```
[ 2.  2.  2.  2.  2.]

 2
 2
 2
 2
 2
[torch.DoubleTensor of size 5]
```

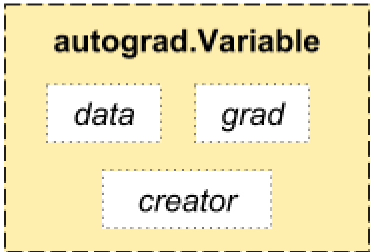
CUDA Tensors

Tensors can be moved onto GPU using the `.cuda` function.

```
# let us run this cell only if CUDA is available
if torch.cuda.is_available():
    x = x.cuda()
    y = y.cuda()
    x + y
```

Autograd : Variable

- autograd package provides automatic differentiation for all operations on Tensors
- Variable



- Create variable

```
import torch
from torch.autograd import Variable
```

Create a variable:

```
x = Variable(torch.ones(2, 2), requires_grad=True)
print(x)
```

Out:

```
Variable containing:
 1  1
 1  1
[torch.FloatTensor of size 2x2]
```

Autograd : Variable (cont'd)

- Addition

```
y = x + 2
print(y)
```

Out:

```
Variable containing:
 3  3
 3  3
[torch.FloatTensor of size 2x2]
```

- multiplication, mean

```
z = y * y * 3
out = z.mean()

print(z, out)
```

Out:

```
Variable containing:
 27  27
 27  27
[torch.FloatTensor of size 2x2]
 Variable containing:
 27
[torch.FloatTensor of size 1]
```

Autograd : Gradients

- In previous example,computation graph is defined as
  - y = x+2
  - z = y*y*3
  - out = z.mean()

- Backpropagation:

➔ out.backward()

- print d(out)/dx

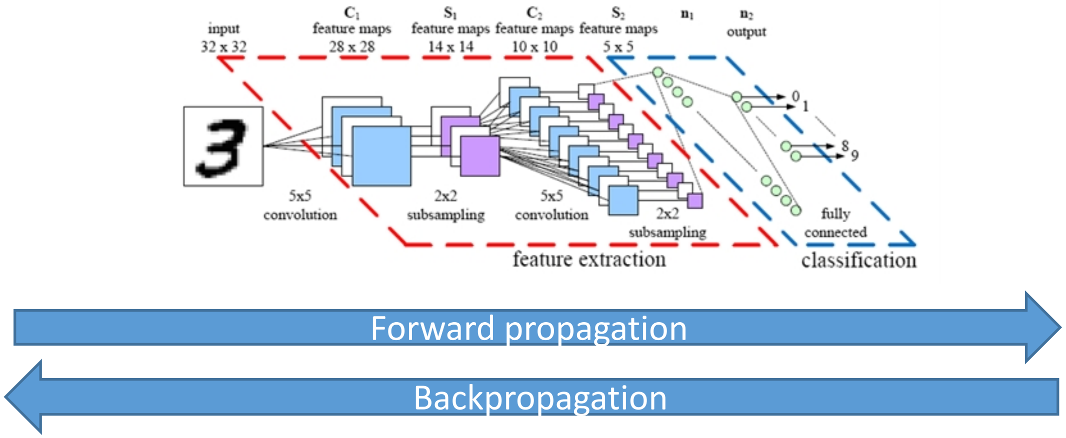
➔ print(x.grad)

Out:

```
Variable containing:
 4.5000  4.5000
 4.5000  4.5000
[torch.FloatTensor of size 2x2]
```

You should have got a matrix of `4.5`. Let's call the `out` *Variable* "o". We have that $o = \frac{1}{4}\sum_i z_i$, $z_i = 3(x_i + 2)^2$ and $z_i\big|_{x_i=1} = 27$. Therefore, $\frac{\partial o}{\partial x_i} = \frac{3}{2}(x_i + 2)$, hence $\frac{\partial o}{\partial x_i}\big|_{x_i=1} = \frac{9}{2} = 4.5$.

Neural Networks : Typical training procedure



1. Define the neural network that has some learnable parameters (or weights)

2. Iterate over a dataset of inputs

3. Process input through the network

4. Compute the loss (how far is the output from being correct)

5. Propagate gradients back into the network's parameters

6. Update the weights of the network, typically using a simple update rule:
   weight = weight - learning_rate * gradient

Define the network

```
import torch
from torch.autograd import Variable
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()
        # 1 input image channel, 6 output channels, 5x5 square convolution
        # kernel
        self.conv1 = nn.Conv2d(1, 6, 5)
        self.conv2 = nn.Conv2d(6, 16, 5)
        # an affine operation: y = Wx + b
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)

    def forward(self, x):
        # Max pooling over a (2, 2) window
        x = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        # If the size is a square you can only specify a single number
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = x.view(-1, self.num_flat_features(x))
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x

    def num_flat_features(self, x):
        size = x.size()[1:]  # all dimensions except the batch dimension
        num_features = 1
        for s in size:
            num_features *= s
        return num_features


net = Net()
```

Define the loss

```
output = net(input)
target = Variable(torch.arange(1, 11))  # a dummy target, for example
criterion = nn.MSELoss()

loss = criterion(output, target)
print(loss)
```

Out:

```
Variable containing:
 38.7810
[torch.FloatTensor of size 1]
```

Training neural networks

```python
import torch
from torch.autograd import Variable

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random Tensors to hold inputs and outputs
x = Variable(torch.randn(N, D_in))
y = Variable(torch.randn(N, D_out), requires_grad=False)

# Use the nn package to define our model and loss
function.
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
)

# The nn package also contains definitions of popular
loss functions; Mean Squared Error (MSE)
loss_fn = torch.nn.MSELoss(size_average=False)

learning_rate = 1e-4
```

```python
for t in range(500):
    # Forward pass
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    print(t, loss.data[0])

    # Zero the gradients before running the backward pass.
    model.zero_grad()

    # Backward pass
     loss.backward()

    # Update the weights using gradient descent
    for param in model.parameters():
        param.data -= learning_rate * param.grad.data
```

Training neural networks with optimizer

```python
import torch
from torch.autograd import Variable

# N is batch size; D_in is input dimension;
# H is hidden dimension; D_out is output dimension.
N, D_in, H, D_out = 64, 1000, 100, 10

# Create random Tensors to hold inputs and outputs
x = Variable(torch.randn(N, D_in))
y = Variable(torch.randn(N, D_out), requires_grad=False)

# Use the nn package to define our model and loss
function.
model = torch.nn.Sequential(
    torch.nn.Linear(D_in, H),
    torch.nn.ReLU(),
    torch.nn.Linear(H, D_out),
)

# The nn package also contains definitions of popular
loss functions; Mean Squared Error (MSE)
loss_fn = torch.nn.MSELoss(size_average=False)

learning_rate = 1e-4
```

```python
optimizer = torch.optim.Adam(model.parameters(),  r=learning_rate)
for t in range(500):
    # Forward pass
    y_pred = model(x)
    loss = loss_fn(y_pred, y)
    print(t, loss.data[0])

    # Use optimizer object to zero all of the
    # gradients for the variables it will update
    optimizer.zero_grad()

    # Backward pass
    loss.backward()

    # Calling the step function on an Optimizer makes an update
to its parameters
    optimizer.step()
```

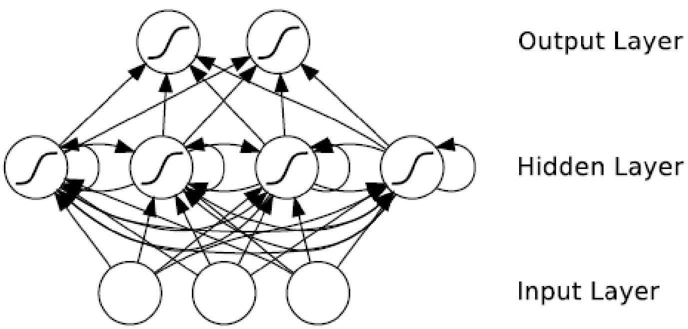
cf) adam optimizer : D.P. Kingma et al., "Adam: A Method for Stochastic Optimization"
https://arxiv.org/abs/1412.6980

Deep Learning based Chatbot

1. RNNs & LSTMs
2. Word embedding vectors and Language models
3. Sequence-to-sequence models
4. Recent deep learning architectures for dialogs

Recurrent Neural Network

- Neural networks whose connections allow **cyclical** connections.



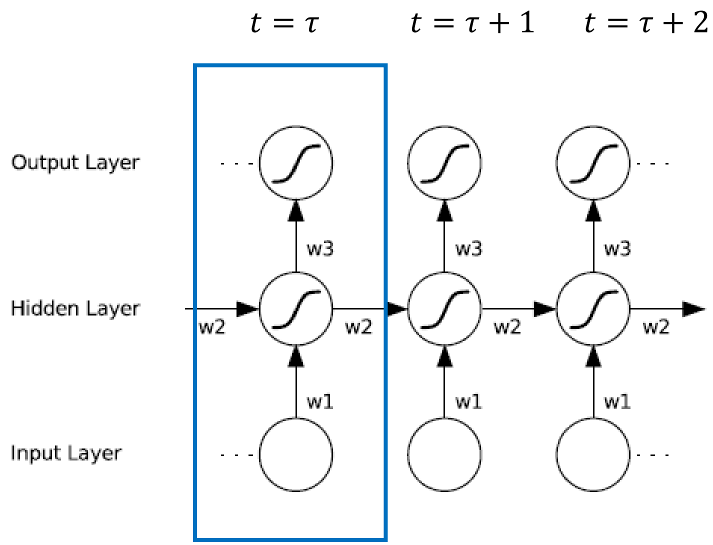
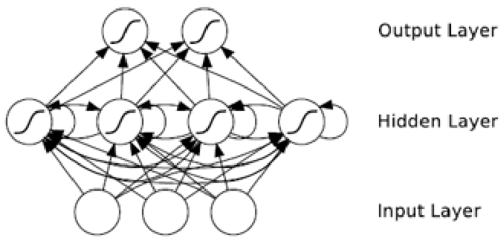
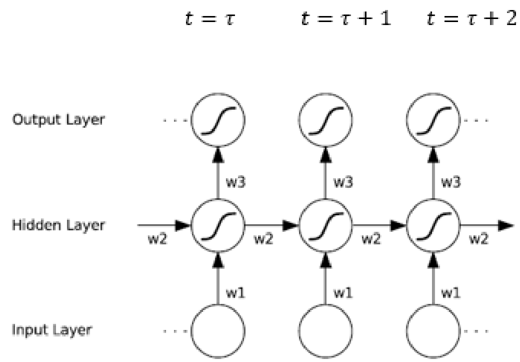


Figure 3.4: **An unfolded recurrent network.** Each node represents a layer of network units at a single timestep. The weighted connections from the input layer to hidden layer are labelled 'w1', those from the hidden layer to itself (i.e. the recurrent weights) are labelled 'w2' and the hidden to output weights are labelled 'w3'. Note that the same weights are reused at every timestep. Bias weights are omitted for clarity.

Recurrent Neural Network





y_k^t

$b_h^t = \theta(a_h^t)$

x_i^t

- RNN with I input units, H hidden units, and K output units.

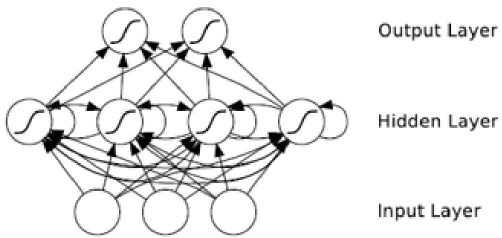
- ***<u>Forward pass</u>***

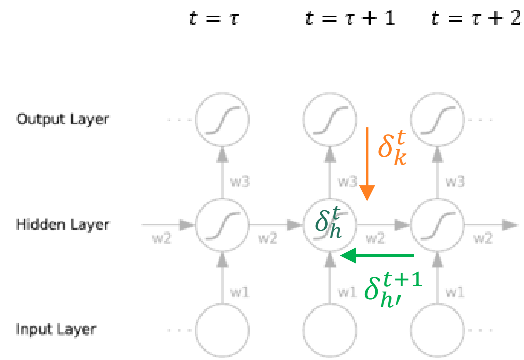
Given an input sequence $x = (\mathbf{x}^1, \ldots, \mathbf{x}^T)$

Recurrent hidden neuron term
: memory of previous input

- x_i^t: the value of input i at time t
- a_h^t: network input to unit h at time t $a_h^t = \sum_{i=1}^{I} w_{ih}^{(1)} x_i^t + \boxed{\sum_{h'=1}^{H} w_{h'h}^{(2)} b_{h'}^{t-1}}$
- b_h^t: the activation of unit h at time t $b_h^t = \theta_h(a_h^t)$
- y_k^t: the value of output k at time t $y_k^t = \sum_{h=1}^{H} w_{hk}^{(3)} b_h^t$

Recurrent Neural Network





y_k^t

$b_h^t = \theta(a_h^t)$

x_i^t

- x_i^t: the value of input i at time t
- a_h^t: network input to unit h at time t $a_h^t = \sum_{i=1}^{I} w_{ih}^{(1)} x_i^t + \sum_{h'=1}^{H} w_{h'h}^{(2)} b_{h'}^{t-1}$
- b_h^t: the activation of unit h at time t $b_h^t = \theta_h(a_h^t)$
- y_k^t: the value of output k at time t $y_k^t = \sum_{h=1}^{H} w_{hk}^{(3)} b_h^t$

- ***Backward pass***
  - Backpropagation through time (BPTT; Williams and Zipser, 1995; Werbos, 1990)
  - Using chain rule, $\mathcal{L}$ differentiable loss function

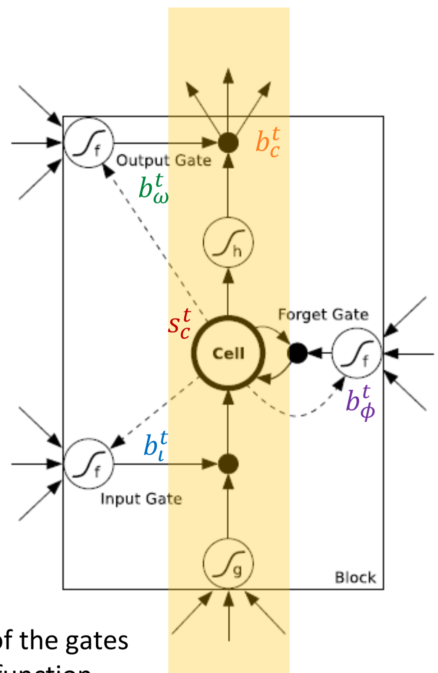
- $\frac{\partial \mathcal{L}}{\partial w_{hk}^{(3)}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial y_k^t} \frac{\partial y_k^t}{\partial w_{hk}^{(3)}} = \sum_{t=1}^{T} \delta_k^t b_h^t$
- $\frac{\partial \mathcal{L}}{\partial w_{h'h}^{(2)}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial a_h^t} \frac{\partial a_h^t}{\partial w_{h'h}^{(2)}} = \sum_{t=1}^{T} \delta_h^t b_{h'}^{t-1}$
- $\frac{\partial \mathcal{L}}{\partial w_{ih}^{(1)}} = \sum_{t=1}^{T} \frac{\partial \mathcal{L}}{\partial a_h^t} \frac{\partial a_h^t}{\partial w_{ih}^{(1)}} = \sum_{t=1}^{T} \delta_h^t x_i^t$

The same weights are reused at every timestep

- $\delta_h^t \coloneqq \frac{\partial \mathcal{L}}{\partial a_h^t}$
- $\delta_h^t = \theta'(a_h^t)\left(\sum_{k=1}^{K} \delta_k^t w_{hk}^{(3)} + \sum_{h'=1}^{H} \delta_{h'}^{t+1} w_{hh'}^{(2)}\right)$
- δ can be calculated by starting at $t = T$, decrementing t at each step.
- ($\delta_h^{T+1} = 0 \; \forall h$, since no error is received from beyond the end of the sequence)

Recurrent Neural Network + LSTM

- Long Short-term Memory (LSTM, Hochreiter and Schmidhuber, 1997)
  - Consists of a set of recurrently connected subnets, known as **memory blocks**
  - Each block contains one or more self-connected memory **cells** and three multiplicative units – the **input**, **output**, **and forget gates.**
  - The three gates control the activation of the cell via multiplications (•)



f : activation function of the gates
g: cell input activation function
h : cell output activation function

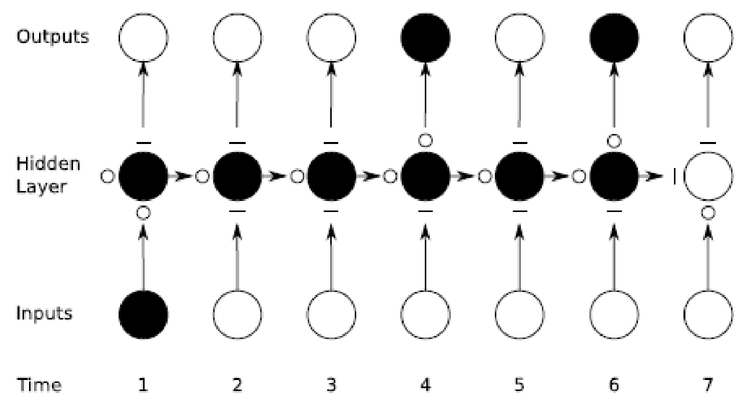
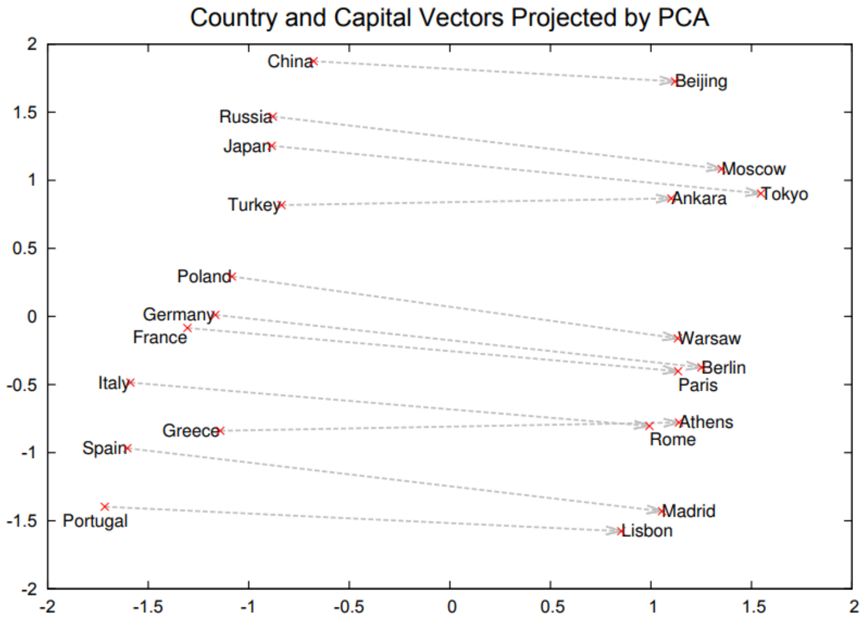
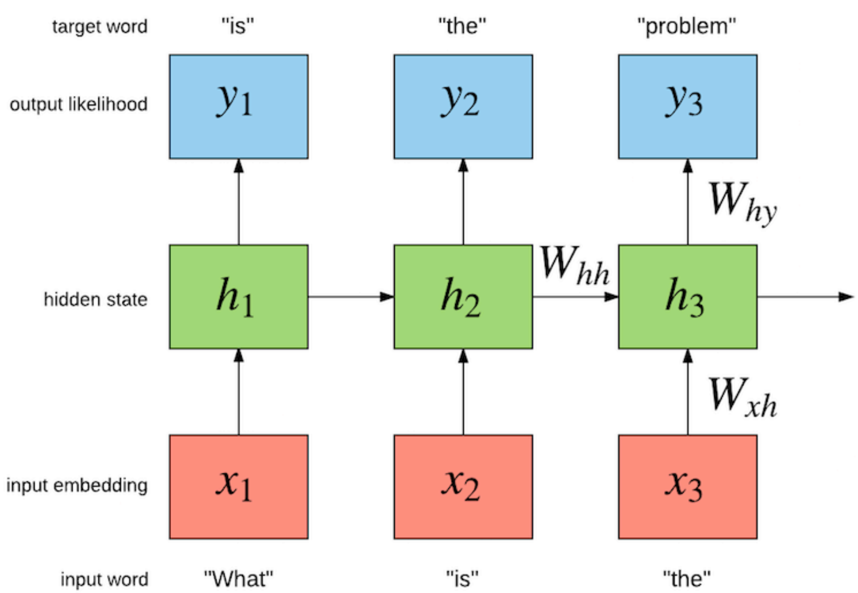


Figure 4.4: **Preservation of gradient information by LSTM.** As in Figure 4.1 the shading of the nodes indicates their sensitivity to the inputs at time one; in this case the black nodes are maximally sensitive and the white nodes are entirely insensitive. The state of the input, forget, and output gates are displayed below, to the left and above the hidden layer respectively. For simplicity, all gates are either entirely open ('O') or closed ('—'). The memory cell 'remembers' the first input as long as the forget gate is open and the input gate is closed. The sensitivity of the output layer can be switched on and off by the output gate without affecting the cell.

Word embedding

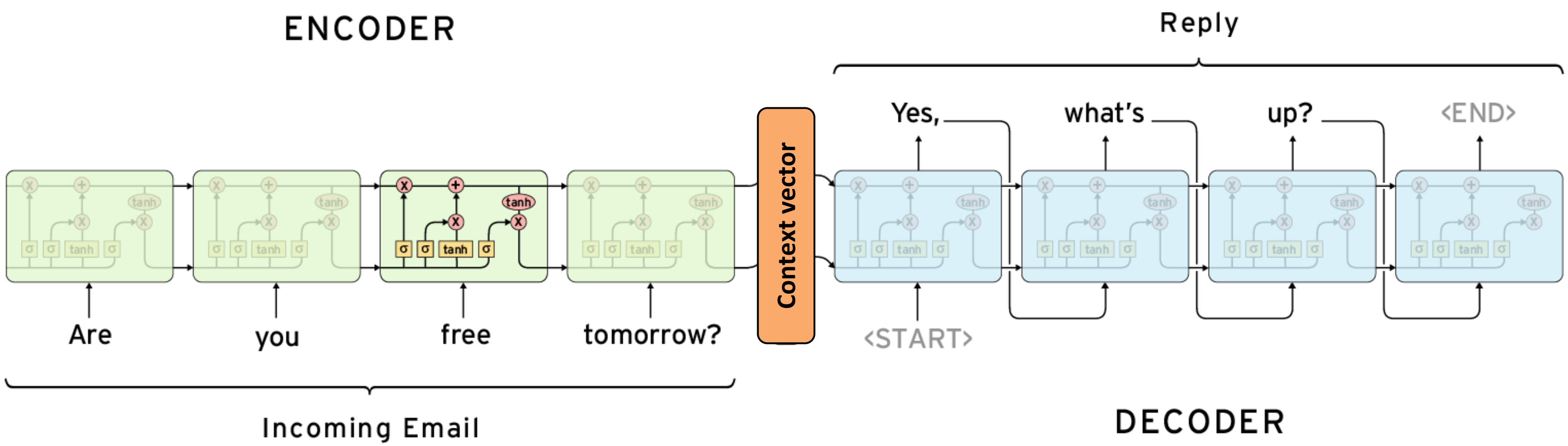




- Every word has its own distributed representation : Word embedding

- ex) RNN Language Model : Word embeddings are fed into the RNN

Sequence to Sequence for Chatbot

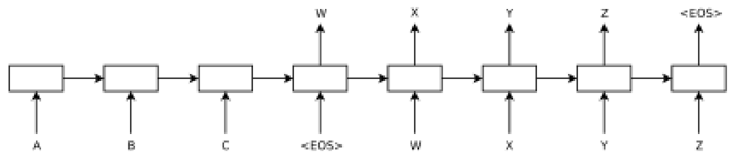


Seq2Seq model comprises of two language models:

- Encoder: encode the input sequence into a fixed length vector (context vector)
- Decoder: generate the next word sequence given the context vector

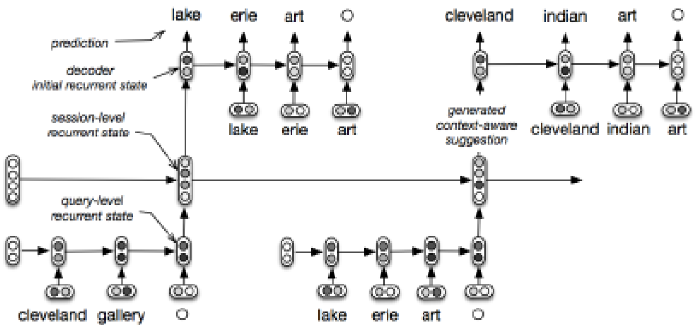
Recent deep learning architectures for dialogs

Sequence to sequence



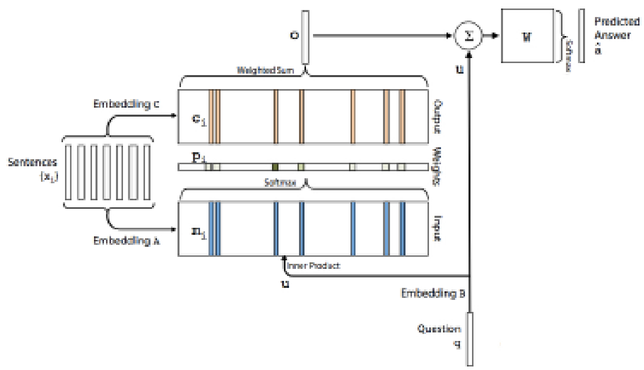
http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Hierarchical recurrent encoder-



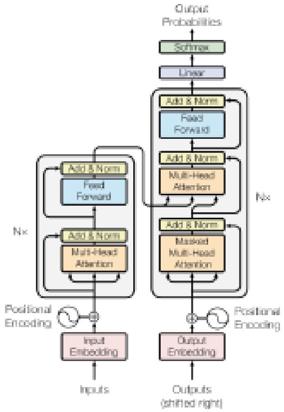
https://arxiv.org/abs/1507.02221

Memory networks



https://arxiv.org/abs/1503.08895

Attention is all you need



https://arxiv.org/abs/1706.03762

ParlAI[par-lay]
: a framework for dialog AI research

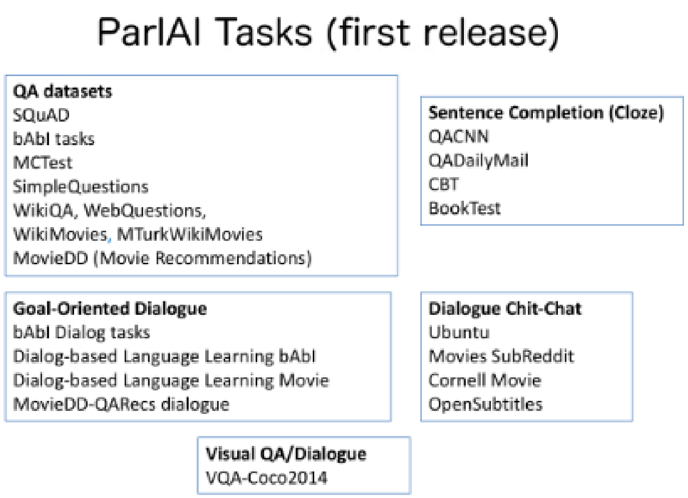
References:

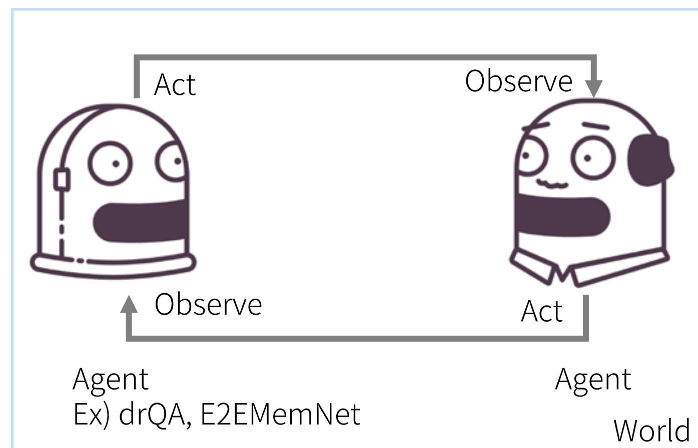
- Alexandr H. Miller et al., ParlAI: A Dialog Research Software Platform, arxiv 2017
- https://github.com/facebookresearch/ParlAI
- http://parl.ai/

ParlAI
Overview

- Open source published by Facebook research
- Its goal is to provide researchers:
  - a unified framework for training and testing dialog models
  - multi-task training over many datasets at once
  - seamless integration of [Amazon Mechanical Turk]() for data collection and human evaluation

- Supporting tasks:





Cf. parley
[NOUN] A parley is a discussion between two opposing people or groups in which both sides try to come to an agreement.
[VERB] When two opposing people or groups parley, they meet to discuss something in order to come to an agreement.

ParlAI: A Dialog Research Software Platform, A. Miller et al., Arxiv, 2017
https://github.com/facebookresearch/ParlAI

ParlAI : Provided tasks

QA datasets
SQuAD, MS MARCO, TriviaQA
bAbI tasks
MCTest
SimpleQuestions
WikiQA, WebQuestions, InsuranceQA
WikiMovies, MTurkWikiMovies
MovieDD (Movie-Recommendations)

Sentence Completion
QACNN
QADailyMail
CBT
BookTest

Dialogue Goal-Oriented
bAbI Dialog tasks, personalized-dialog
Dialog-based Language Learning bAbI
Dialog-based Language Learning Movie
MovieDD-QARecs dialogue)

Dialogue Chit-Chat
Ubuntu
Movies SubReddit
Cornell Movie
OpenSubtitles

VQA/Visual Dialogue
VQA-v1,
VQA-v2, Vdialog, CLEVR

Add your own dataset!
Open source...

ParlAI : Provided tasks

QA datasets
bAbI tasks
MCTest
SquAD, NewsQA, MS MARCO
SimpleQuestions
WebQues
WikiMovi
MovieDD

Dialogue
bAbI Dial
Camrest
Dialog-ba
MovieDD
CommAI-

Sentence Completion
QACNN
QADailyMail
CBT

Attention during mem lookup

Story (2: 2 supporting facts)	Support	Hop 1	Hop 2	Hop 3
John dropped the milk.		0.06	0.00	0.00
John took the milk there.	yes	0.88	1.00	0.00
Sandra went back to the bathroom.		0.00	0.00	0.00
John moved to the hallway.	yes	0.00	0.00	1.00
Mary went back to the bedroom.		0.00	0.00	0.00
Where is the milk? Answer: hallway Prediction: hallway				

Story (18: size reasoning)	Support	Hop 1	Hop 2	Hop 3
The suitcase is bigger than the chest.	yes	0.00	0.88	0.00
The box is bigger than the chocolate.		0.04	0.05	0.10
The chest is bigger than the chocolate.	yes	0.17	0.07	0.90
The chest fits inside the container.		0.00	0.00	0.00
The chest fits inside the box.		0.00	0.00	0.00
Does the suitcase fit in the chocolate? Answer: no Prediction: no				

ParlAI : Provided tasks

QA datasets
bAbI tasks
MCTest
SquAD, NewsQA, MS MA
SimpleQuestions
WebQuestions, WikiQA
WikiMovies, MTurkWikiM
MovieDD (Movie-Recomm

Dialogue Goal-Oriented
bAbI Dialog tasks
Camrest
Dialog-based Language L
MovieDD (QA,Recs dialog
CommAI-env

VQA/Visual Di
TBD..

In meteorology, precipitation is any product of the condensation of atmospheric water vapor that falls under **gravity**. The main forms of precipitation include drizzle, rain, sleet, snow, **graupel** and hail... Precipitation forms as smaller droplets coalesce via collision with other rain drops or ice crystals **within a cloud**. Short, intense periods of rain in scattered locations are called "showers".

What causes precipitation to fall?
gravity

What is another main form of precipitation besides drizzle, rain, snow, sleet and hail?
graupel

Where do water droplets collide with ice crystals to form precipitation?
within a cloud

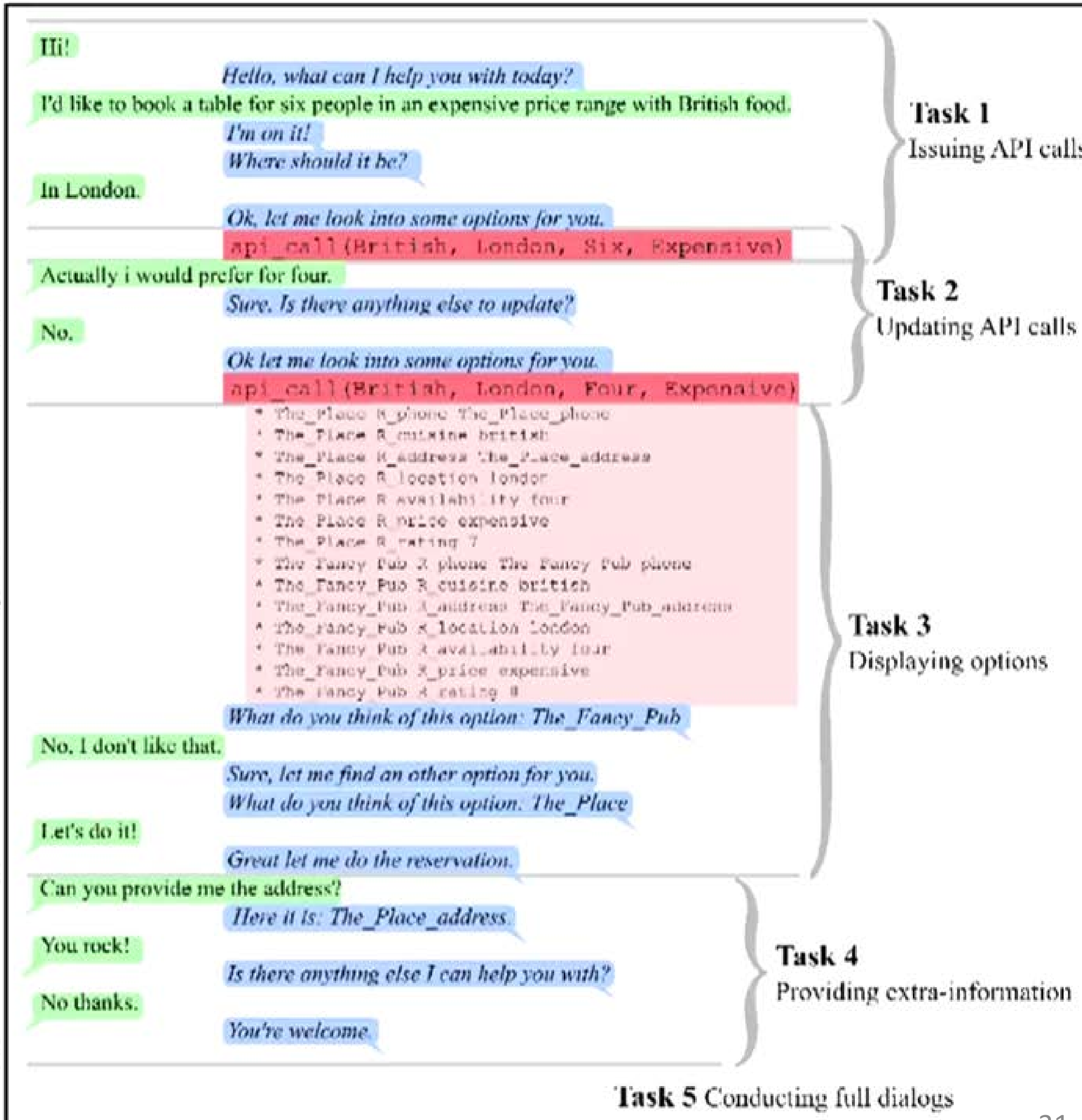
Figure 1: Question-answer pairs for a sample passage in the SQuAD dataset. Each of the answers is a segment of text from the passage.

ParlAI : Provided tasks

QA datasets
bAbI tasks
MCTest
SquAD, NewsQA, MS MARC
SimpleQuestions
WebQuestions, WikiQA
WikiMovies, MTurkWikiMov
MovieDD (Movie-Recommen

Dialogue Goal-Oriented
bAbI Dialog tasks
Camrest
Dialog-based Language Lear
MovieDD (QA,Recs dialogue
CommAI-env

VQA/Visual Dialo
TBD..



ParlAI : how to run examples

- Display 10 random examples from the SQuAD dataset:

python examples/display_data.py -t squad

- Displays 100 random examples from multi-tasking on the bAbI + SQuAD datasets at the same time:

python examples/display_data.py -t babi,squad -n 100

- Evaluate on the bAbI test set with a human agent (using the local keyboard as input):

python examples/eval_model.py -m local_human -t babi -dt valid

- Train an attentive LSTM model on the SQuAD dataset with a batch size of 32 examples:

python examples/train_model.py -m drqa -t squad -bs 32 -mf /tmp/model

Practices

Practice

- Reference : O Vinyals et al., "A Neural Conversational Model", ICML Deep learning workshop, 2016
- Database : Opensubtitle2016 (62M turns)

U: how are you ?
S: good .

U: it 's too hot !
S: i need a shower .

U: do you mind ?
S: yes , i do mind .

U: i was thinking about getting into the boxing .
S: great , why not ?

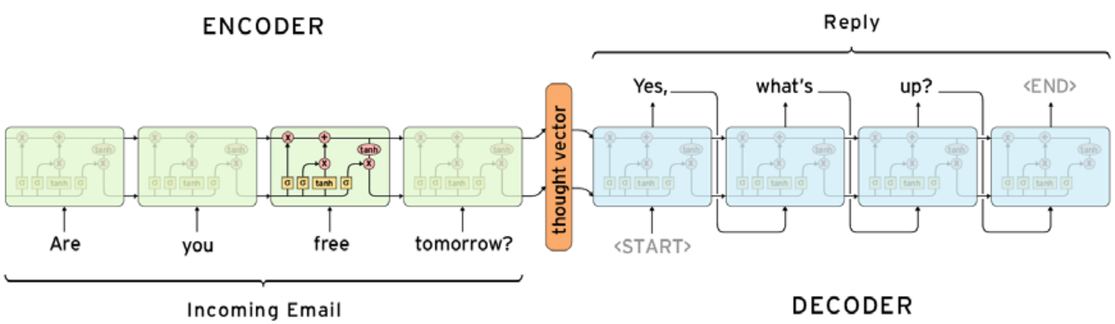
Example : Chitchat

U: and i wish to hell i knew why .
S: when we got back to earth everybody treated us like heroes .

U: master yoshida !
S: the princess is also missing !

Example : lack of context

- Model : RNN Encoder Decoder



Practice

- Prob1. Display opensubtitle data and briefly explain about the database. (10pts)

- Prob2. Fill in the 7 blanks (each 5pts) in parlai/agents/seq2seq/seq2seq_FILL_IN_THE_BLANK.py

(the blanks are with the comment '# FILL HERE') and run the script. (35pts)

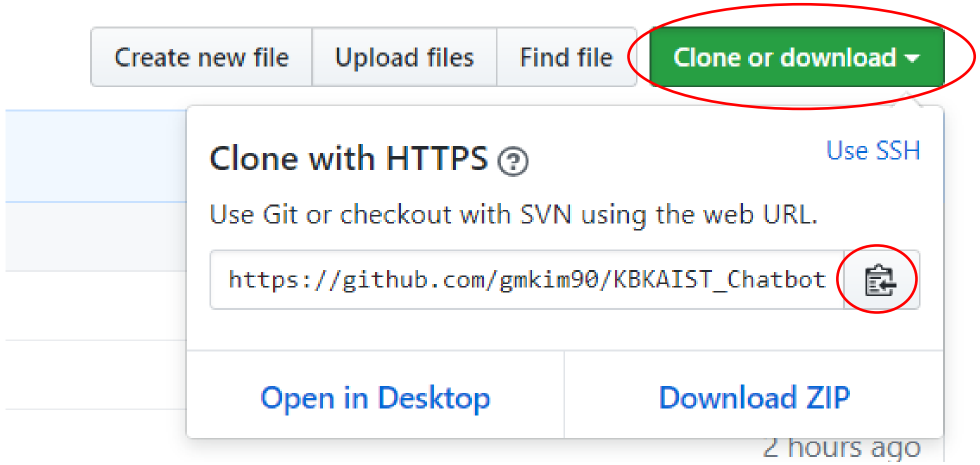
python examples/train_model_practice.py -m seq2seq -t opensubtitles -bs 32 -mf 'data/OpenSubtitles'

- Prob3. Propose idea to improve this model. (30pts)

프로젝트: 챗봇시스템 구현

- 준비: https://github.com/gmkim90/KBKAIST_Chatbot 에 접속

- 준비된 프로젝트 다운받기
  $ git clone https://github.com/gmkim90/KBKAIST_Chatbot.git



- Python3 이용
  $ source activate python3

- ParlAI setup
  $ cd KBKAIST_Chatbot
  $ python setup.py develop

프로젝트: 챗봇시스템 구현

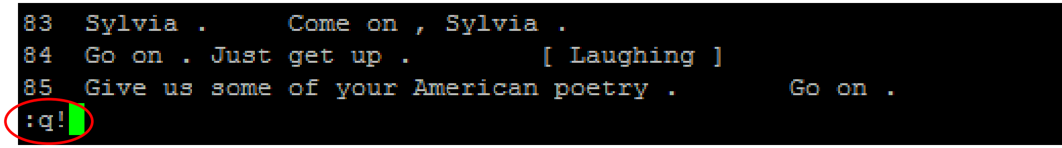
❖ **Prob 1. Display opensubtitle data and briefly explain about the database (10pts)**

$ python examples/display_data.py -t Opensubtitles

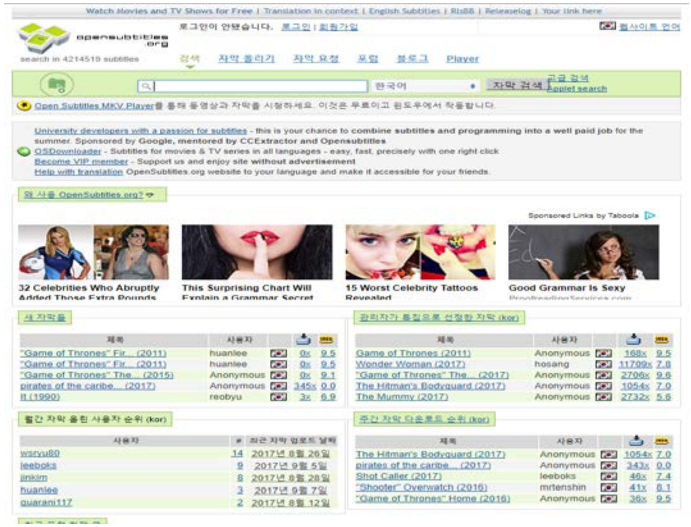
Or

$ vim data/OpenSubtitles/{train, valid, test}.txt

- Close vi
  :q!



- **Opensubtitles**:
  - A collection of document from http://www.opensubtitles.org/
  - Each line represents two turns of a conversation.

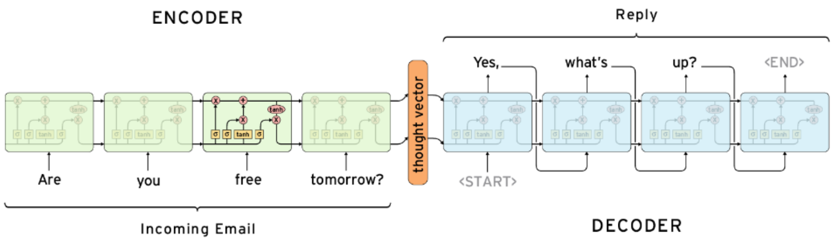


프로젝트: 챗봇시스템 구현

❖ **Prob 2. Fill in *seven* blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- **Run the code**

$ python examples/train_model.py -m seq2seq -t opensubtitles -bs 32 -mf data/OpenSubtitles/model



$ vim parlay/agents/seq2seq/seq2seq.py

```
        # set up modules
        self.criterion = nn.NLLLoss()
        # lookup table stores word embeddings
1.      self.lt = nn.Embedding() # FILL HERE
        # encoder captures the input text
2.      self.encoder = nn.GRU() # FILL HERE
        # decoder produces our output states
3.      self.decoder = nn.GR
        # linear layer helps
4.      self.h2o = nn.Linear
        # dropout on the line
        self.dropout = nn.Dr
        # softmax maps outpu
        self.softmax = nn.Lo
```

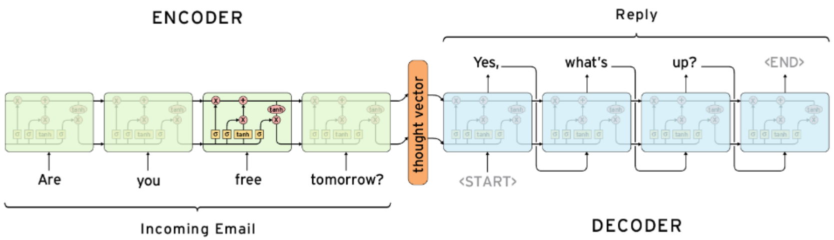
```
    def hidden_to_idx(self, hidden, dropout=False):
        """Converts hidden state vectors into indices into the dictionary."""
        if hidden.size(0) > 1:
            raise RuntimeError('bad dimensions of tensor:', hidden)
        hidden = hidden.squeeze(0)
5.      scores =  # FILL HERE
        if dropout:
6.          scores = # FILL HERE
7.      scores = # FILL HERE
        _max_score, idx = scores.max(1)
        return idx, scores
```

프로젝트: 챗봇시스템 구현

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')

- **1. fill here**



- self.it = nn.Embedding(?)

  Import torch.nn

- Using http://pytorch.org/docs/master/

```
class torch.nn.Embedding(num_embeddings, embedding_dim, padding_idx=None, max_norm=None, norm_type=2, scale_grad_by_freq=False, sparse=False)    [source]
```

프로젝트: 챗봇시스템 구현

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')

- **1. fill here**

- self.it = nn.Emb

  Import torch.nn

- Using http://py

```
class torch.nn.
max_norm=None
```

Parameters:

- num_embeddings (*int*) – size of the dictionary of embeddings
- embedding_dim (*int*) – the size of each embedding vector
- padding_idx (*int, optional*) – If given, pads the output with zeros whenever it encounters the index.
- max_norm (*float, optional*) – If given, will renormalize the embeddings to always have a norm lesser than this
- norm_type (*float, optional*) – The p of the p-norm to compute for the max_norm option
- scale_grad_by_freq (*boolean, optional*) – if given, this will scale gradients by the frequency of the words in the mini-batch.

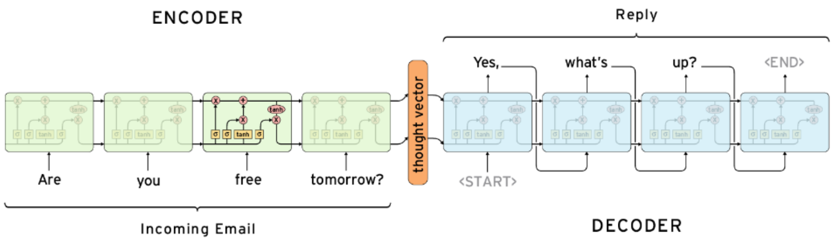
- **Answer:**
  self.it = nn.Embedding(len(self.dict), hsz, padding_idx=self.NULL_IDX, scale_grad_by_freq=True)

프로젝트: 챗봇시스템 구현

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')

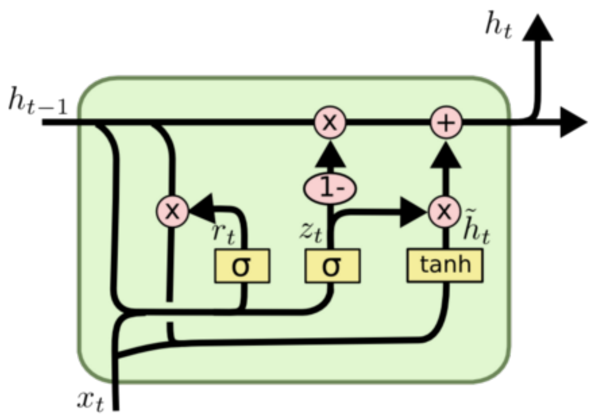
- **2, 3. fill here**



- self.encoder = nn.GRU(?)
- self.decoder = nn.GRU(?)

- Using http://pytorch.org/docs/master/

class `torch.nn.GRU` (**args, **kwargs*) [source]



$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$

$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$

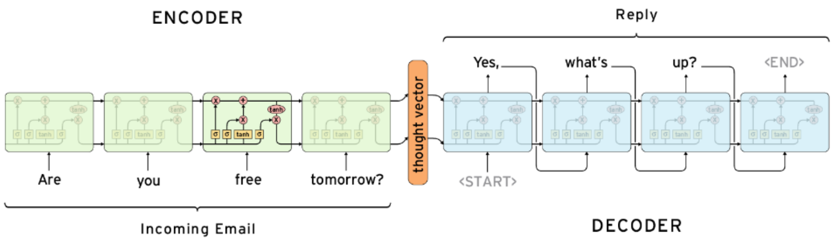
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')

- **2, 3. fill here**



- self.encoder = nn.GRU(?)
- self.decoder = nn.GRU(?)

- Using http://pytorch.org/docs/master/

```
class torch.nn.GRU(*args, **kwargs)    [source]
```

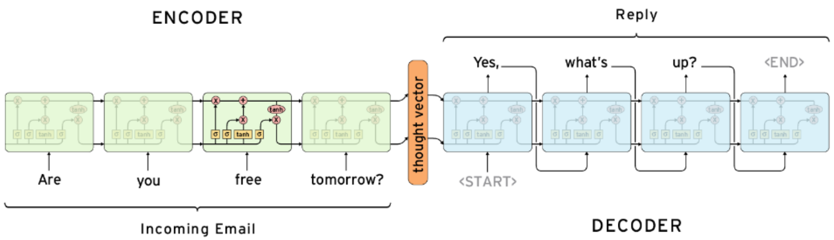
- **Answer**:
  self.encoder = nn.GRU(hsz, hsz, opt['numlayers'])
  self.decoder = nn.GRU(hsz, hsz, opt['numlayers'])

프로젝트: 챗봇시스템 구현

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')

- **4. fill here**



- self.h2o = nn.Linear(?)

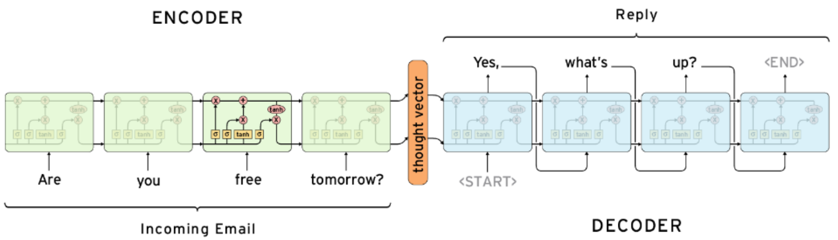
- Using http://pytorch.org/docs/master/

```
class torch.nn.Linear(in_features, out_features, bias=True)    [source]
```

- **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')

- **4. fill here**



- self.h2o = nn.Linear(?)

- Using http://pytorch.org/docs/master/

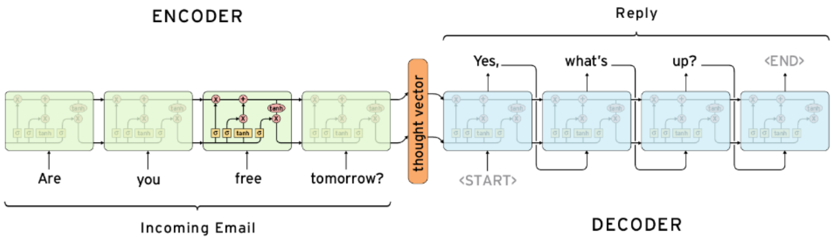
class torch.nn.Linear(*in_features, out_features, bias=True*) [source]

- **Answer**:
  self.h2o = nn.Linear(hsz, len(self.dict))

프로젝트: 챗봇시스템 구현

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type '/# FILL HERE')



- **5~7. fill here**

```
    def hidden_to_idx(self, hidden, dropout=False):
        """Converts hidden state vectors into indices into the dictionary."""
        if hidden.size(0) > 1:
            raise RuntimeError('bad dimensions of tensor:', hidden)
        hidden = hidden.squeeze(0)
5.      scores =  # FILL HERE
6.      if dropout:
            scores = # FILL HERE
7.      scores = # FILL HERE
        _max_score, idx = scores.max(1)
        return idx, scores
```

❖ **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

- # FILL HERE
- (if you want to find the location, type ‘/# FILL HERE’)

- **5~7. fill here**
- **Under** def hidden_to_idx(self, hidden, dropout=False):

- **Answer:**

```
scores = self.h2o(hidden)
If dropout:
        scores = self.dropout(scores)
scores = self.softmax(scores)
```

프로젝트: 챗봇시스템 구현

- **Prob 2. Fill in the blanks (each 5 pts) of parlai/agents/seq2seq/seq2seq.py and run (35pts)**

Run the code now

python examples/train_model.py -m seq2seq -t opensubtitles -bs 32 -mf 'data/OpenSubtitles'

- **Prob 3. Propose idea to improve this model (30pts)**