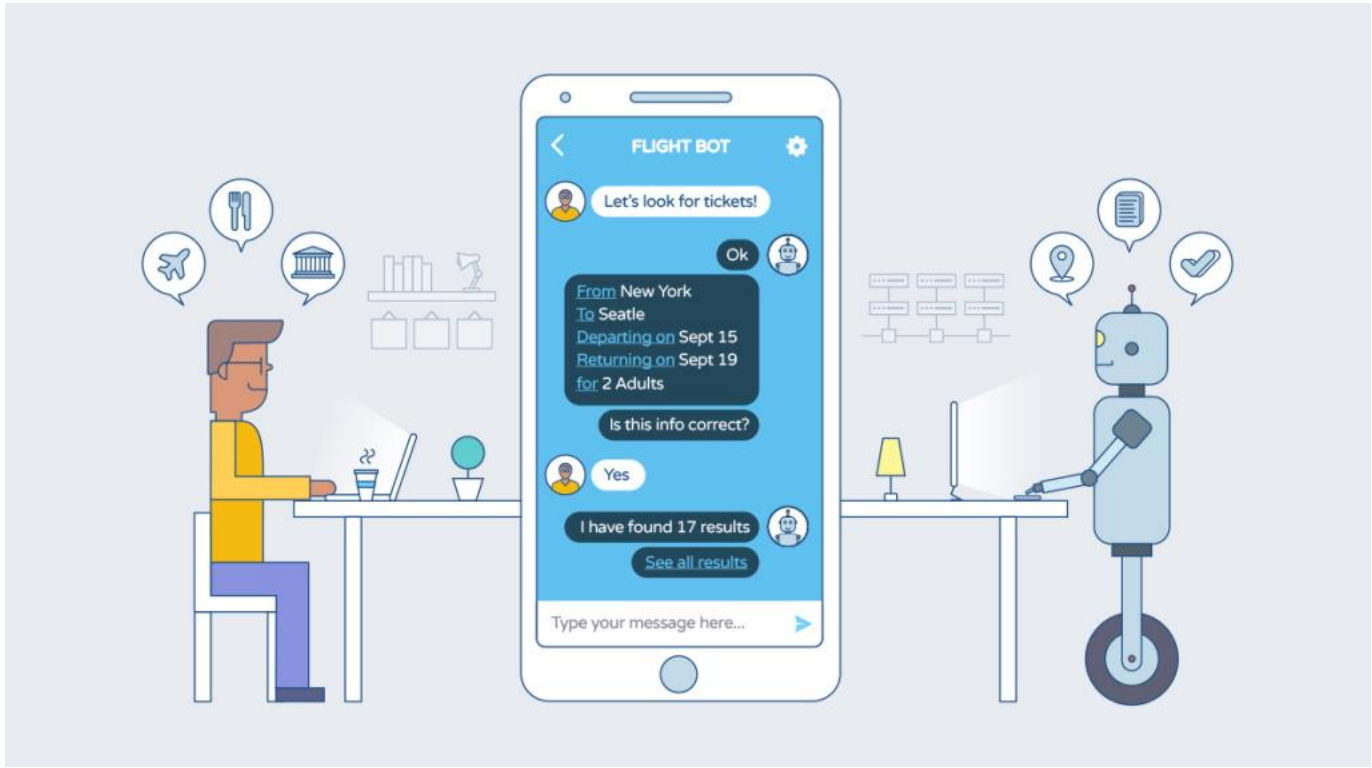
Seq2seq Chatbot Modeling

RNN 조 : 박세승 박수빈 이승연 정승연

Contents

Motivation



나 너무 힘들다

그래도 해야지

개강이다

방학이 참 짧죠

Seq2seq 모델을 이용한 일상 대화 챗봇 모델링

Dataset

간단한 기초 대화

18	가상화폐 폭딱 망함	어서 잊고 새출발 하세요.
19	가스불 켜고 나갔어	빨리 집에 돌아가서 끄고 나오세요.
20	가스불 켜놓고 나온거 같아	빨리 집에 돌아가서 끄고 나오세요.
21	가스비 너무 많이 나왔다.	다음 달에는 더 절약해봐요.
22	가스비 비싼데 감기 걸리겠	따뜻하게 사세요!

감정 공감에 집중

내가 이 회사를 다니는 게 맞나... 그런 생각도 들고요.	걱정이 되는 그 마음 이해해요. 멋진 미래를 꿈꾸기에는 세상이 너무 팍팍한 것 같아요.
앞으로 집은 어떻게 사나 싶어요.	언젠가는 나아질 거라는 믿음이 사람을 버티게 만들어주는 것 같아요.
좋은 환경에 있어도 잘될까 말까 하는데... 이런 환경에서 사람답게 살 수 있을까?	계속 앞으로 가다 보면 당신에게 맞는 길도 찾을 수 있지 않을까요?
너무 답답해요.	안 좋은 상황이신 거 같아요. 무슨 일인지 저에게 자세히 얘기해줄 수 있을까요?
너무 답답해서 미칠 거 같을 때가 있어요.	많이 답답하시겠어요. 오늘만큼은 천천히 산책이라도 해보는 게 어때요?
눈앞이 깜깜할 때도 있어요.	그랬군요. 제가 당신의 마음을 조금이라도 이해할 수 있으면 좋겠어요.

실제 질환자와의 대화

위염인 줄 알았는데 암이었다는 사람도 있었고... 걱정이 되지.	병원에 가서 검진을 받아보면 어떨까요?
간암도 조기에는 잘 모른다고 하니까... 내가 간암일 수도 있지 않나, 내가 간암이면 어쩌나 그런 걱정도 해.	혹여나 그렇다고 하더라도 조기에 발견하면 완치할 수 있을 거예요.
암이면 어떻게 하지?	정기적인 검진이 암을 예방하는 데에 큰 도움이 된대요.

Preprocessing

```
<PAD>
<SOS>
<END>
<UNK>
생일이야
한고
환심을
차단한
집앞
말아보셔요
삶도
서류에서부터
로봇도
```

```
input_variable: tensor([[ 8182, 16354,  4045,  1822,  8747],
        [  440,  7706,    54,  1082,    64],
        [   39,    17,  4041,    17,     2],
        [  431,     9,     2,     2,     0],
        [ 1746,     2,     0,     0,     0],
        [    2,     0,     0,     0,     0]])
lengths: tensor([6, 5, 4, 4, 3])
target_variable: tensor([[4851, 9855,  186, 1917, 1329],
        [4852, 1024, 4046, 1936,  276],
        [  68,  201,  130,  633,    9],
        [   9,    9,    9, 1548,    2],
        [   2,    2,    2, 2742,    0],
        [   0,    0,    0,    9,    0],
        [   0,    0,    0,    2,    0]])
mask: tensor([[1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 1],
        [1, 1, 1, 1, 0],
        [0, 0, 0, 1, 0],
        [0, 0, 0, 1, 0]], dtype=torch.uint8)
max_target_len: 7
```

전처리 과정

1. 물음표, 마침표 등의 중요하지 않은 **문장 부호 제거**

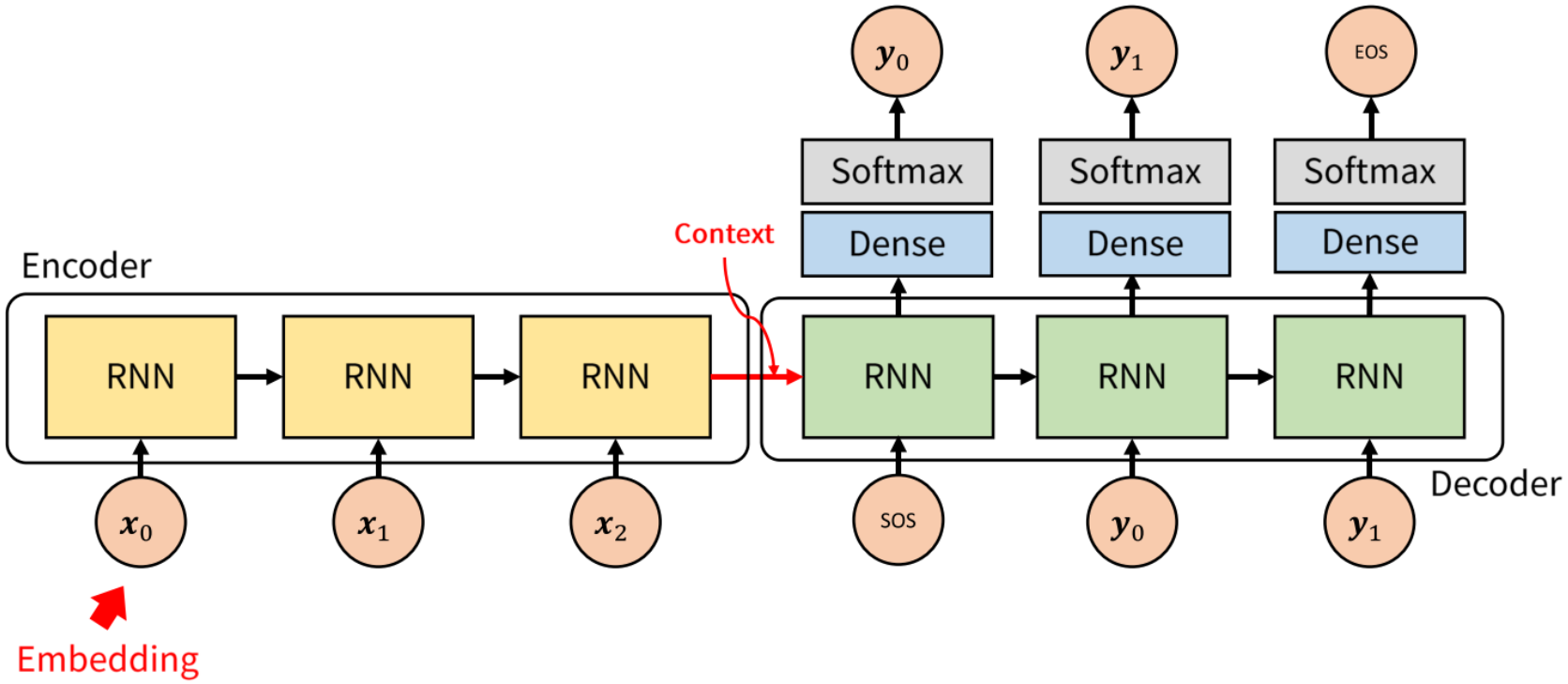
2. 단어를 띄어쓰기 기준으로 **토큰화**

3. **빈출 단어** 위주로 단어집을 만들어서 **정수값 부여** 및 텐서 변환

4. 사이즈를 같게 만들어 주기 위해 짧은 단어에는 **zero padding** 추가 + 너무 긴 문장 버리도록 **MAX_LEN** 지정

5. padding 부분이 train loss 에 추가 되지 않도록 **Masking** 및 **문장의 시작과 끝을 구분**해주도록 특수 토큰 부여

Modeling seq2seq

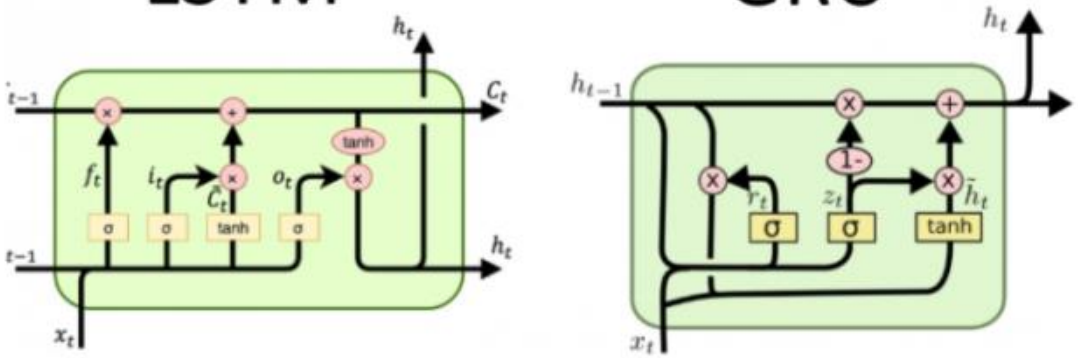


Encoder ➪ 순차적으로 입력 받은 단어정보를 압축해서 context vector 생성 ➪ Decoder ➪ 단어를 한 개씩 출력

Decoder의 초기 입력으로 **문장의 시작**을 의미하는 심볼 <sos>가 들어가고
다음 단어 **예측**은 **끝**을 의미하는 심볼인 <eos>가 다음 단어로 예측될 때까지

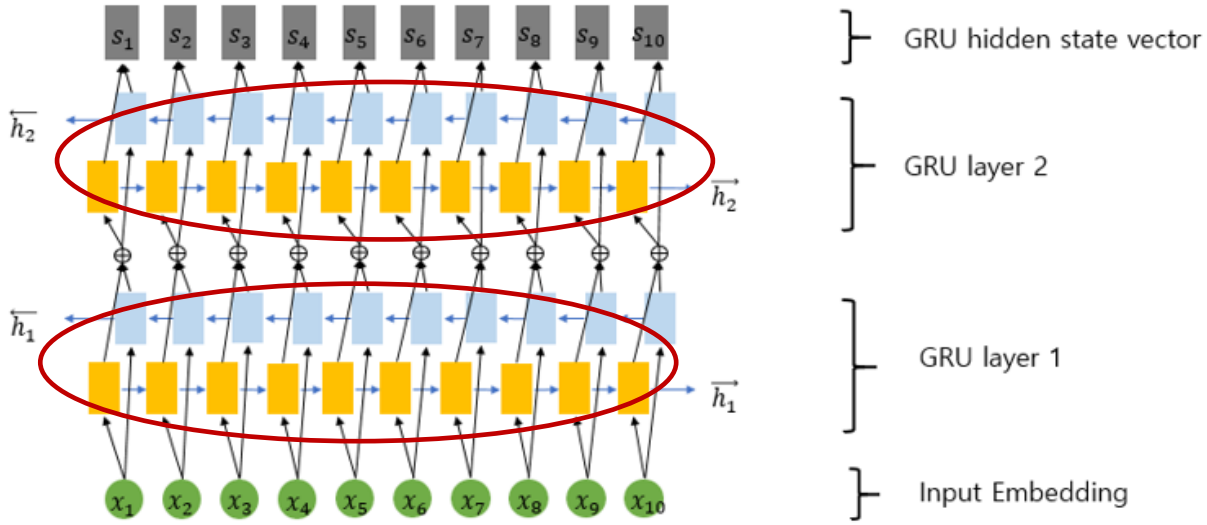
Modeling

seq2seq – Encoder



LSTM vs GRU network

GRU unit이 LSTM보다 parameter 수가 적다



학습에 사용한 encoder model

양방향 GRU를 두 layer 쌓아서 과거, 미래 모두의 문맥 고려

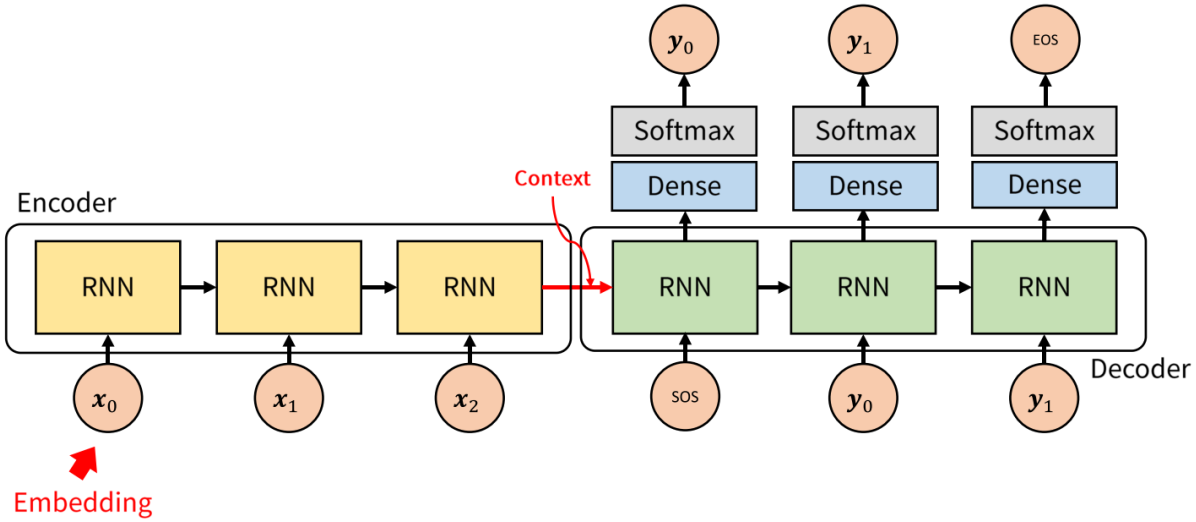
Modeling

seq2seq – Encoder

```
class EncoderRNN(nn.Module):
    def __init__(self, hidden_size, embedding, n_layers=1, dropout=0):
        super(EncoderRNN, self).__init__()
        self.n_layers = n_layers
        self.hidden_size = hidden_size
        self.embedding = embedding
         self.gru = nn.GRU(hidden_size, hidden_size, n_layers,
                           dropout=(0 if n_layers == 1 else dropout), bidirectional=True)

    def forward(self, input_seq, input_lengths, hidden=None):
        # 단어 인덱스를 임베딩으로 변환합니다
        embedded = self.embedding(input_seq)
        # RNN 모듈을 위한 패딩된 배치 시퀀스를 패킹합니다
        packed = nn.utils.rnn.pack_padded_sequence(embedded, input_lengths)
        # GRU로 포워드 패스를 수행합니다
        outputs, hidden = self.gru(packed, hidden)
        # 패딩을 언패킹합니다
        outputs, _ = nn.utils.rnn.pad_packed_sequence(outputs)
        # 양방향 GRU의 출력을 합산합니다
        outputs = outputs[:, :, :self.hidden_size] + outputs[:, : ,self.hidden_size:]
        # 출력과 마지막 은닉 상태를 반환합니다
        return outputs, hidden
```

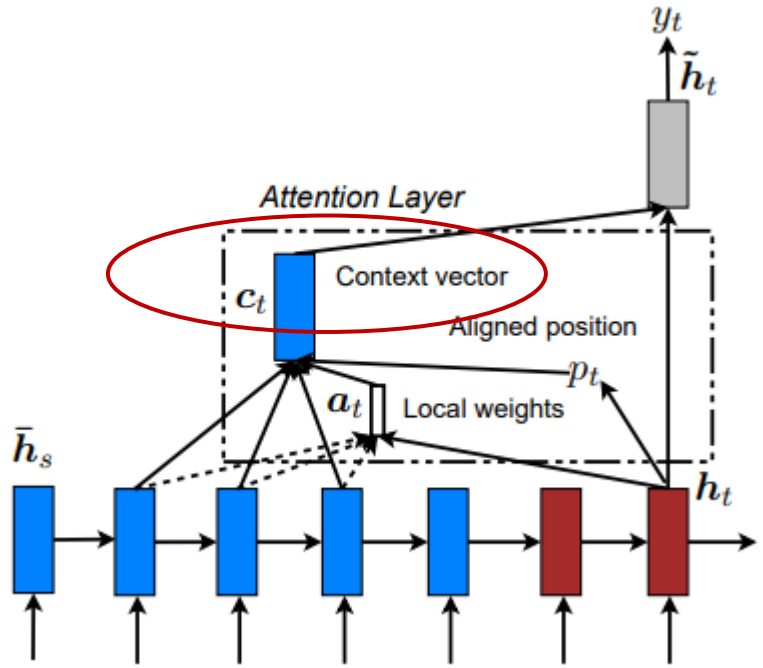
Modeling

seq2seq – Attention



기존의 seq2seq model

문장의 길이가 길어지면 단순히 요약된 context vector
만으로는 출력 생성에 어려움이 있음



Attention model (local attention)

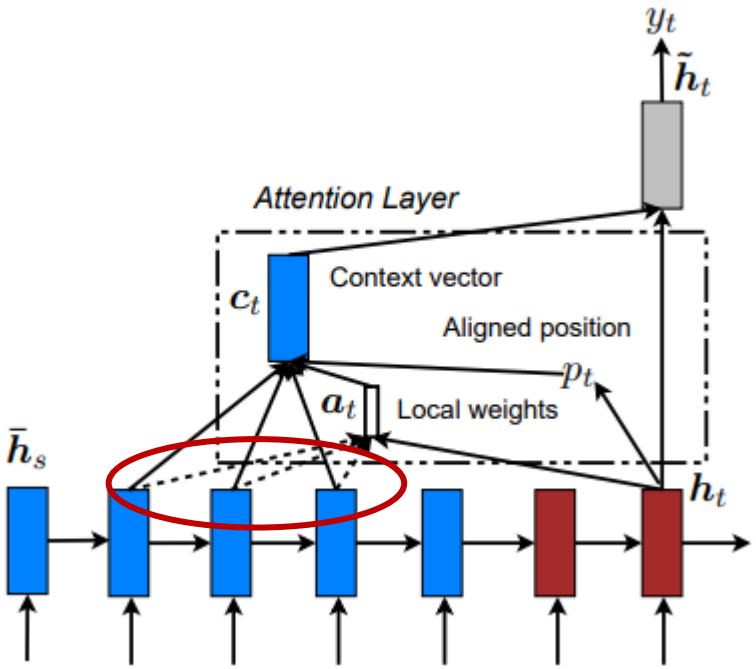
$$p_t = S \cdot \text{sigmoid}(\boldsymbol{v}_p^\top \tanh(\boldsymbol{W}_p \boldsymbol{h}_t))$$

$$\boldsymbol{a}_t(s) = \text{align}(\boldsymbol{h}_t, \bar{\boldsymbol{h}}_s) \exp\left(-\frac{(s - p_t)^2}{2\sigma^2}\right)$$

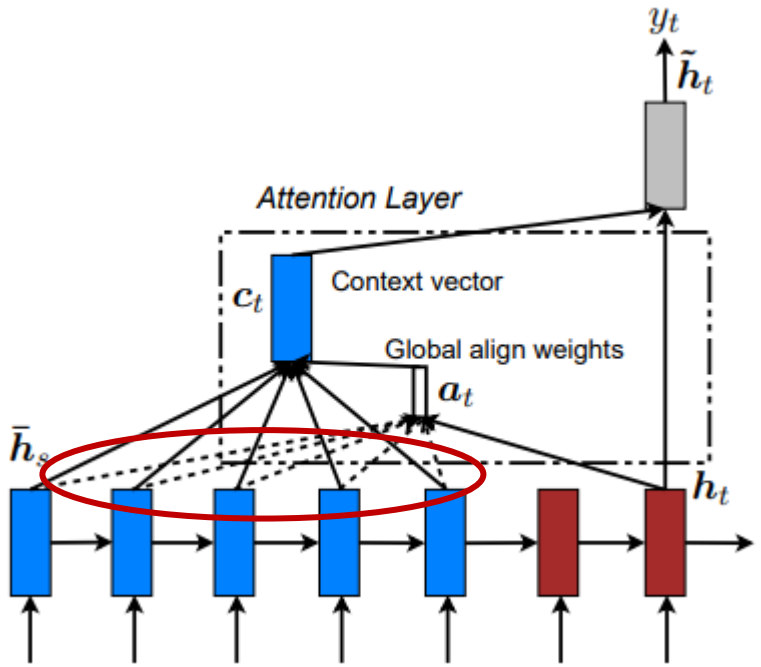
Modeling

seq2seq – Attention

https://arxiv.org/pdf/1508.04025.pdf



Local Attention



Global Attention

$$a_t(s) = \text{align}(h_t, \bar{h}_s) = \frac{\exp\left(\text{score}(h_t, \bar{h}_s)\right)}{\sum_{s'} \exp\left(\text{score}(h_t, \bar{h}_{s'})\right)}$$

$$\text{score}(h_t, \bar{h}_s) = \begin{cases} h_t^\top \bar{h}_s & \textit{dot} \\ h_t^\top W_a \bar{h}_s & \textit{general} \\ v_a^\top \tanh\left(W_a[h_t; \bar{h}_s]\right) & \textit{concat} \end{cases}$$

$$c_i = \sum_{j=1}^{n} \alpha_{i,j} h_j$$

Modeling

seq2seq – Attention

```
class Attn(nn.Module):
    def __init__(self, method, hidden_size):
        super(Attn, self).__init__()
        self.method = method
        if self.method not in ['dot', 'general', 'concat']:
            raise ValueError(self.method, "is not an appropriate attention method.")
        self.hidden_size = hidden_size
        if self.method == 'general':
            self.attn = nn.Linear(self.hidden_size, hidden_size)
        elif self.method == 'concat':
            self.attn = nn.Linear(self.hidden_size * 2, hidden_size)
            self.v = nn.Parameter(torch.FloatTensor(hidden_size))


    def dot_score(self, hidden, encoder_output):
        return torch.sum(hidden * encoder_output, dim=2)

    def general_score(self, hidden, encoder_output):
        energy = self.attn(encoder_output)
        return torch.sum(hidden * energy, dim=2)

    def concat_score(self, hidden, encoder_output):
        energy = self.attn(torch.cat((hidden.expand(encoder_output.size(0), -1, -1), encoder_output), 2)).tanh()
        return torch.sum(self.v * energy, dim=2)
```

Modeling

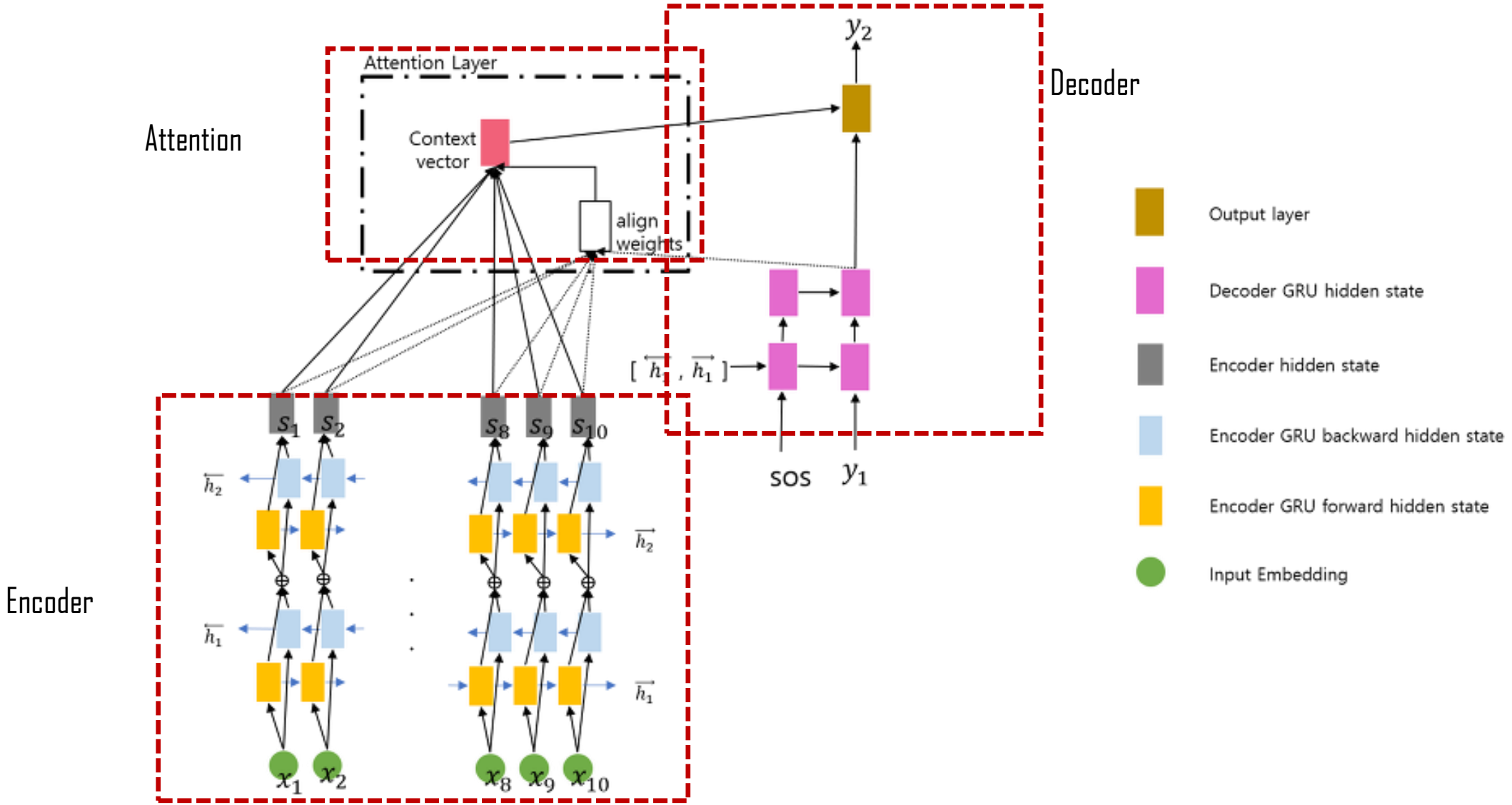
seq2seq – Decoder

```
class LuongAttnDecoderRNN(nn.Module):
    def __init__(self, attn_model, embedding, hidden_size, output_size, n_layers=1, dropout=0.1):
        super(LuongAttnDecoderRNN, self).__init__()

        # 참조를 보존해 둡니다
        self.attn_model = attn_model
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.n_layers = n_layers
        self.dropout = dropout

        # 레이어를 정의합니다
        self.embedding = embedding
        self.embedding_dropout = nn.Dropout(dropout)
        self.gru = nn.GRU(hidden_size, hidden_size, n_layers, dropout=(0 if n_layers == 1 else dropout))
        self.concat = nn.Linear(hidden_size * 2, hidden_size)
        self.out = nn.Linear(hidden_size, output_size)
        self.attn = Attn(attn_model, hidden_size)
```

Modeling

```
def forward(self, input_step, last_hidden, encoder_outputs):
        embedded = self.embedding(input_step)
        embedded = self.embedding_dropout(embedded)
        rnn_output, hidden = self.gru(embedded, last_hidden)
        attn_weights = self.attn(rnn_output, encoder_outputs)
        context = attn_weights.bmm(encoder_outputs.transpose(0, 1))
        rnn_output = rnn_output.squeeze(0)
        context = context.squeeze(1)
        concat_input = torch.cat((rnn_output, context), 1)
        concat_output = torch.tanh(self.concat(concat_input))
        output = self.out(concat_output)
        output = F.softmax(output, dim=1)

        return output, hidden
```

Modeling


Attention
Attention Layer
Context vector
align weights
Decoder
y_2
y_1
SOS
$[\overleftarrow{h}_1, \overrightarrow{h}_1]$
Encoder
s_1 s_2 s_8 s_9 s_{10}
$\overleftarrow{h}_2$
$\overleftarrow{h}_1$
$\overrightarrow{h}_2$
$\overrightarrow{h}_1$
x_1 x_2 x_8 x_9 x_{10}
Output layer
Decoder GRU hidden state
Encoder hidden state
Encoder GRU backward hidden state
Encoder GRU forward hidden state
Input Embedding

Modeling *results*

- **Loss**: 다양한 단어들에 대해 확률을 계산하여 Cross Entropy 사용
- **Optimizer**: Adam
- **Hyperparameters**
  - Hidden Size 256
  - Batch Size 128
  - Dropout 0.2
  - Learning Rate 0.001
  - Decoder Ratio 10
  - Max Length 8

Modeling *results*

> 화가 나
Bot: 무슨 일이 있으셨나요 ? 일이 있나요 ? 것 같아요 .
> 너무 슬퍼
Bot: 많이 힘드시겠어요 . 더 자세히 말씀해 주시겠어요 ?

> 화가 나
Bot: 그럴수도 저녁은 . . 저녁은 그렇게 게 좋을 . . .
> 너무 슬프고 걱정돼
Bot: 그 재미가 저녁은 생각을 재미가 재미가 어떨까요 ?
> 뭐라고?
Bot: 다시 더 돌면 내면 . . . 좋아하시나봐요 . .
> 그래..
Bot: 그 . . . 세워봐요 . 세워봐요 . 세워봐요 .
> 짜증난다고
Bot: 다시 한달이 해보세요 . . . . . .

> 자꾸 화가 나
Bot: 무슨 일이 있으셨나요 ? 더 자세히 말씀해 주실 수 있나요 ?
> 슬퍼
Bot: 많이 힘든가봅니다 . 잘 하세요 . . . . .
> 일이 생겨서 좀 걱정돼
Bot: 무슨 일이 있으셨나요 ? 더 자세히 말씀해 주실 수 있나요 ?

Discussion

모델 특성

- 문장 길이에 제한을 주고 긴 문장을 효과적으로 다룰 수 있는 attention 이용
- 문장이 짧은 경우, LSTM 및 GRU 등의 단순 RNN으로도 좋은 성능을 낼 수 있음

모델의 한계점 및 개선방안

- 베이스로 둔 모델이 영어 기반 데이터에 적합했음
- 한국어는 단순 띄어쓰기 보다, 형태소 분석을 통해 토큰화를 하는 것이 더욱 효과적일 듯
- 데이터셋의 특성마다 결과가 상이하게 나옴 (오버피팅 된 듯)

THANK YOU

Q&A