파이토치 튜토리얼:
TRANSLATION WITH A SEQUENCE TO SEQUENCE NETWORK AND ATTENTION

Taebong Moon

Feb 13th 2019

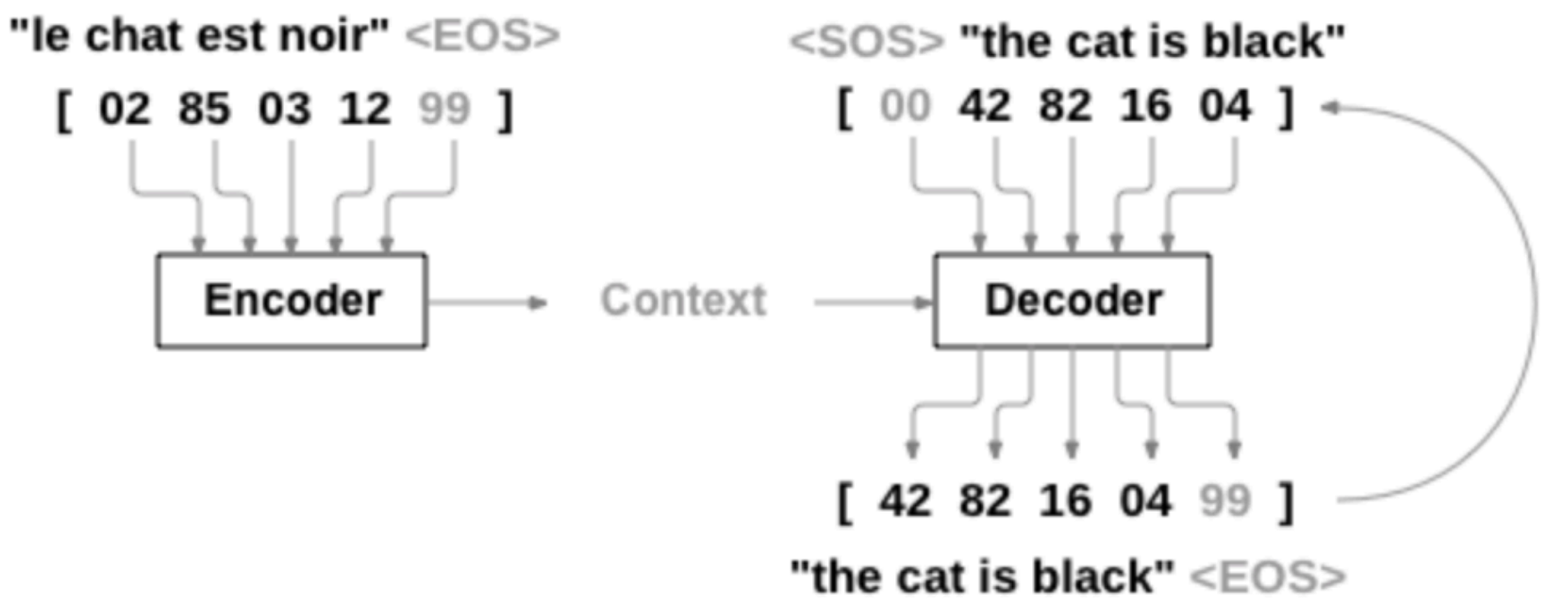
목차

1. 목표

```
[KEY: > input, = target, < output]

> il est en train de peindre un tableau .
= he is painting a picture .
< he is painting a picture .

> pourquoi ne pas essayer ce vin delicieux ?
= why not try that delicious wine ?
< why not try that delicious wine ?

> elle n est pas poete mais romanciere .
= she is not a poet but a novelist .
< she not not a poet but a novelist .

> vous etes trop maigre .
= you re too skinny .
< you re all alone .
```

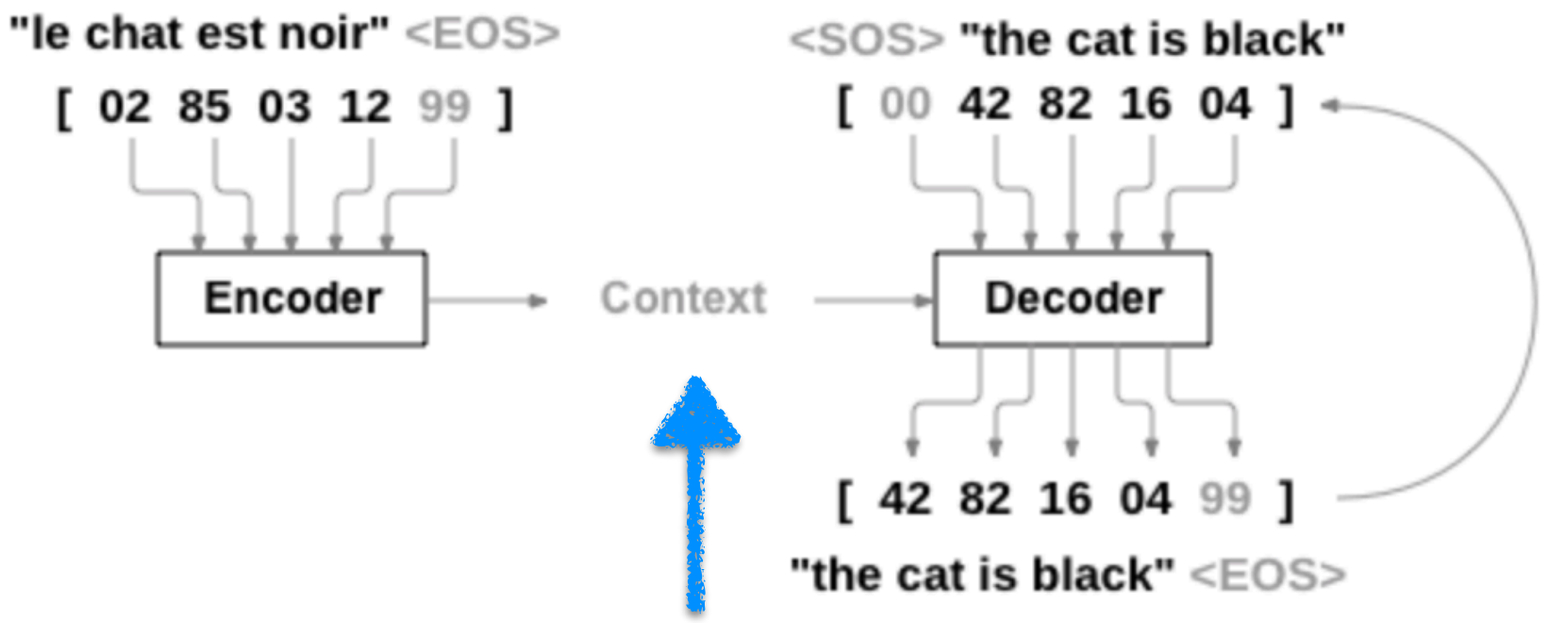
프랑스어에서 영어로 번역하는 모델 학습

Seq to seq network



- 두 개의 Recurrent Neural Network
- Encoder: condense an input sequence into a vector (fixed context)
- Decoder: unfolds that vector into a new sequence

Seq to seq network



전체 input seq.를 encoding하기엔 부담 (burden). -> Attention

- 두 개의 Recurrent Neural Network
- Encoder: condense an input sequence into a vector (fixed context)
- Decoder: unfolds that vector into a new sequence

2. 파일 읽기와 전처리

Requirements

```
from __future__ import unicode_literals, print_function, division
from io import open
import unicodedata
import string
import re
import random

import torch
import torch.nn as nn
from torch import optim
import torch.nn.functional as F

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
```

2. 파일 읽기와 전처리

```
Go.	Va !
Run!	Cours !
Run!	Courez !
Wow!	Ça alors !
Fire!	Au feu !
Help!	À l'aide !
Jump.	Saute.
Stop!	Ça suffit !
Stop!	Stop !
Stop!	Arrête-toi !
```

- 데이터셋: eng-fra.txt
- english to french pair

```
135839	Death is something that we're often discouraged to talk about or even think about, but I've realized that preparing for death is one of the most empowering things you can do. Thinking about death clarifies your life.	La mort est une chose qu'on nous décourage souvent de discuter ou même de penser mais j'ai pris conscience que se préparer à la mort est l'une des choses que nous puissions faire qui nous investit le plus de responsabilité. Réfléchir à la mort clarifie notre vie.
135840	Since there are usually multiple websites on any given topic, I usually just click the back button when I arrive on any webpage that has pop-up advertising. I just go to the next page found by Google and hope for something less irritating.	Puisqu'il y a de multiples sites web sur chaque sujet, je clique d'habitude sur le bouton retour arrière lorsque j'atterris sur n'importe quelle page qui contient des publicités surgissantes. Je me rends juste sur la prochaine page proposée par Google et espère tomber sur quelque chose de moins irritant.
135841	If someone who doesn't know your background says that you sound like a native speaker, it means they probably noticed something about your speaking that made them realize you weren't a native speaker. In other words, you don't really sound like a native speaker.	Si quelqu'un qui ne connaît pas vos antécédents dit que vous parlez comme un locuteur natif, cela veut dire qu'il a probablement remarqué quelque chose à propos de votre élocution qui l'a fait prendre conscience que vous n'êtes pas un locuteur natif. En d'autres termes, vous ne parlez pas vraiment comme un locuteur natif.
135842	It may be impossible to get a completely error-free corpus due to the nature of this kind of collaborative effort. However, if we encourage members to contribute sentences in their own languages rather than experiment in languages they are learning, we might be able to minimize errors.	Il est peut-être impossible d'obtenir un Corpus complètement dénué de fautes, étant donnée la nature de ce type d'entreprise collaborative. Cependant, si nous encourageons les membres à produire des phrases dans leurs propres langues plutôt que d'expérimenter dans les langues qu'ils apprennent, nous pourrions être en mesure de réduire les erreurs.
135843
```

2. 파일 읽기와 전처리

```
SOS_token = 0
EOS_token = 1


class Lang:
    def __init__(self, name):
        self.name = name
        self.word2index = {}
        self.word2count = {}
        self.index2word = {0: "SOS", 1: "EOS"}
        self.n_words = 2  # Count SOS and EOS

    def addSentence(self, sentence):
        for word in sentence.split(' '):
            self.addWord(word)

    def addWord(self, word):
        if word not in self.word2index:
            self.word2index[word] = self.n_words
            self.word2count[word] = 1
            self.index2word[self.n_words] = word
            self.n_words += 1
        else:
            self.word2count[word] += 1
```

- helper class: Lang
- 네트워크 input과 target으로 사용하기 위한 단어당 unique index 필요

2. 파일 읽기와 전처리

```
# Turn a Unicode string to plain ASCII, thanks to
# https://stackoverflow.com/a/518232/2809427
def unicodeToAscii(s):
    return ''.join(
        c for c in unicodedata.normalize('NFD', s)
        if unicodedata.category(c) != 'Mn'
    )

# Lowercase, trim, and remove non-letter characters
```

```
def normalizeString(s):
    s = unicodeToAscii(s.lower().strip())
    s = re.sub(r"([.!?])", r" \1", s)
    s = re.sub(r"[^a-zA-Z.!?]+", r" ", s)
    return s
```

- Unicode 문자를 ASCII로 변경
- 대문자 -> 소문자, Punctuation 다듬기

2. 파일 읽기와 전처리

```
def readLangs(lang1, lang2, reverse=False):
    print("Reading lines...")

    # Read the file and split into lines
    lines = open('data/%s-%s.txt' % (lang1, lang2), encoding='utf-8').\
        read().strip().split('\n')

    # Split every line into pairs and normalize
    pairs = [[normalizeString(s) for s in l.split('\t')] for l in lines]

    # Reverse pairs, make Lang instances
    if reverse:
        pairs = [list(reversed(p)) for p in pairs]
        input_lang = Lang(lang2)
        output_lang = Lang(lang1)
    else:
        input_lang = Lang(lang1)
        output_lang = Lang(lang2)

    return input_lang, output_lang, pairs
```

- 파일 읽기 -> into lines -> into pairs
- reverse를 활용하여 영어 -> 프랑스어, 프랑스어 -> 영어로 번역 가능

2. 파일 읽기와 전처리

```
MAX_LENGTH = 10

eng_prefixes = (
    "i am ", "i m ",
    "he is", "he s ",
    "she is", "she s ",
    "you are", "you re ",
    "we are", "we re ",
    "they are", "they re "
)
```

```
def filterPair(p):
    return len(p[0].split(' ')) < MAX_LENGTH and \
        len(p[1].split(' ')) < MAX_LENGTH and \
        p[1].startswith(eng_prefixes)


def filterPairs(pairs):
    return [pair for pair in pairs if filterPair(pair)]
```

- 빠른 학습을 위해 짧고 단순한 문장 사용 (최대 단어 길이 = 10)
- I am, he is 등등으로 시작하는 문장 선택

2. 파일 읽기와 전처리

기술

파이썬 startswith () 메소드는, 그렇지 않은 경우는 false는 true를 반환하면 문자열, 시작 부분에 서브 문자열을 지정 여부를 확인하는 데 사용됩니다. 인수가 구걸하고 지정된 값을 종료하는 경우, 지정된 범위 내에서 확인합니다.

문법

startswith () 메서드 구문 :

```
str.startswith(str, beg=0,end=len(string));
```

매개 변수

- STR - 문자열을 발견했습니다.
- strbeg는 - 선택적 매개 변수는 문자열 검색의 시작 위치를 설정하는 데 사용됩니다.
- strend - 선택적 매개 변수는 문자열의 끝 위치 감지를 설정하는 데 사용됩니다.

반환 값

이 발견되면 문자열 true, 그렇지 않은 경우는 false 반환됩니다.

startswith()

예

다음의 예는 startswith ()를 사용하는 기능을 보여줍니다

```
#!/usr/bin/python

str = "this is string example....wow!!!";
print str.startswith( 'this' );
print str.startswith( 'is', 2, 4 );
print str.startswith( 'this', 2, 4 );
```

다음, 상기 출력 결과의 예 :

```
True
True
False
```

2. 파일 읽기와 전처리

```
def prepareData(lang1, lang2, reverse=False):
    input_lang, output_lang, pairs = readLangs(lang1, lang2, reverse)
    print("Read %s sentence pairs" % len(pairs))
    pairs = filterPairs(pairs)
    print("Trimmed to %s sentence pairs" % len(pairs))
    print("Counting words...")
    for pair in pairs:
        input_lang.addSentence(pair[0])
        output_lang.addSentence(pair[1])
    print("Counted words:")
    print(input_lang.name, input_lang.n_words)
    print(output_lang.name, output_lang.n_words)
    return input_lang, output_lang, pairs


input_lang, output_lang, pairs = prepareData('eng', 'fra', True)
print(random.choice(pairs))
```

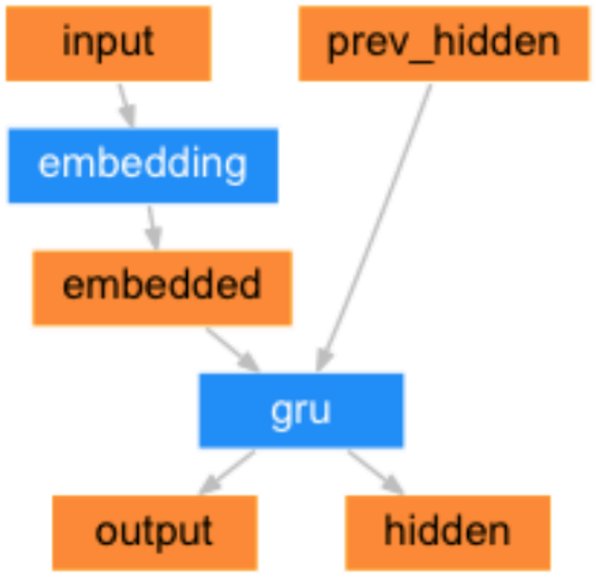
Out:

```
Reading lines...
Read 135842 sentence pairs
Trimmed to 10599 sentence pairs
Counting words...
Counted words:
fra 4345
eng 2803
['je ne suis pas gros .', 'i m not fat .']
```

- 파일 읽기 -> into lines -> into pairs
- Normalize text, filter by length and content
- 단어장 만들기

3. Seq to seq 모델

Encoder

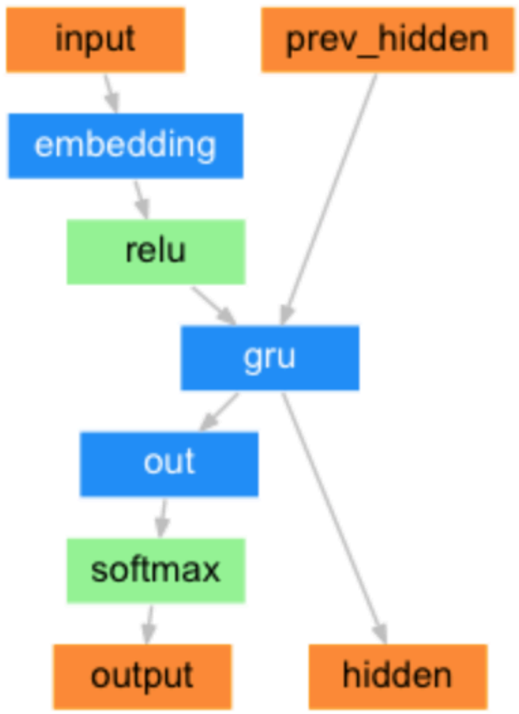



```
class EncoderRNN(nn.Module):
    def __init__(self, input_size, hidden_size):
        super(EncoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding = nn.Embedding(input_size, hidden_size)
        self.gru = nn.GRU(hidden_size, hidden_size)

    def forward(self, input, hidden):
        embedded = self.embedding(input).view(1, 1, -1)
        output = embedded
        output, hidden = self.gru(output, hidden)
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

- **output, h_n = GRU(input, h_0)**
  - input: (seq_len, batch, input_size)
  - h_0: (num_layers*num_directions, batch, hidden_size)
  - output: (seq_len, batch, num_directions*hidden_size)
  - h_n: (num_layers * num_directions, batch, hidden_size)

3. Seq to seq 모델

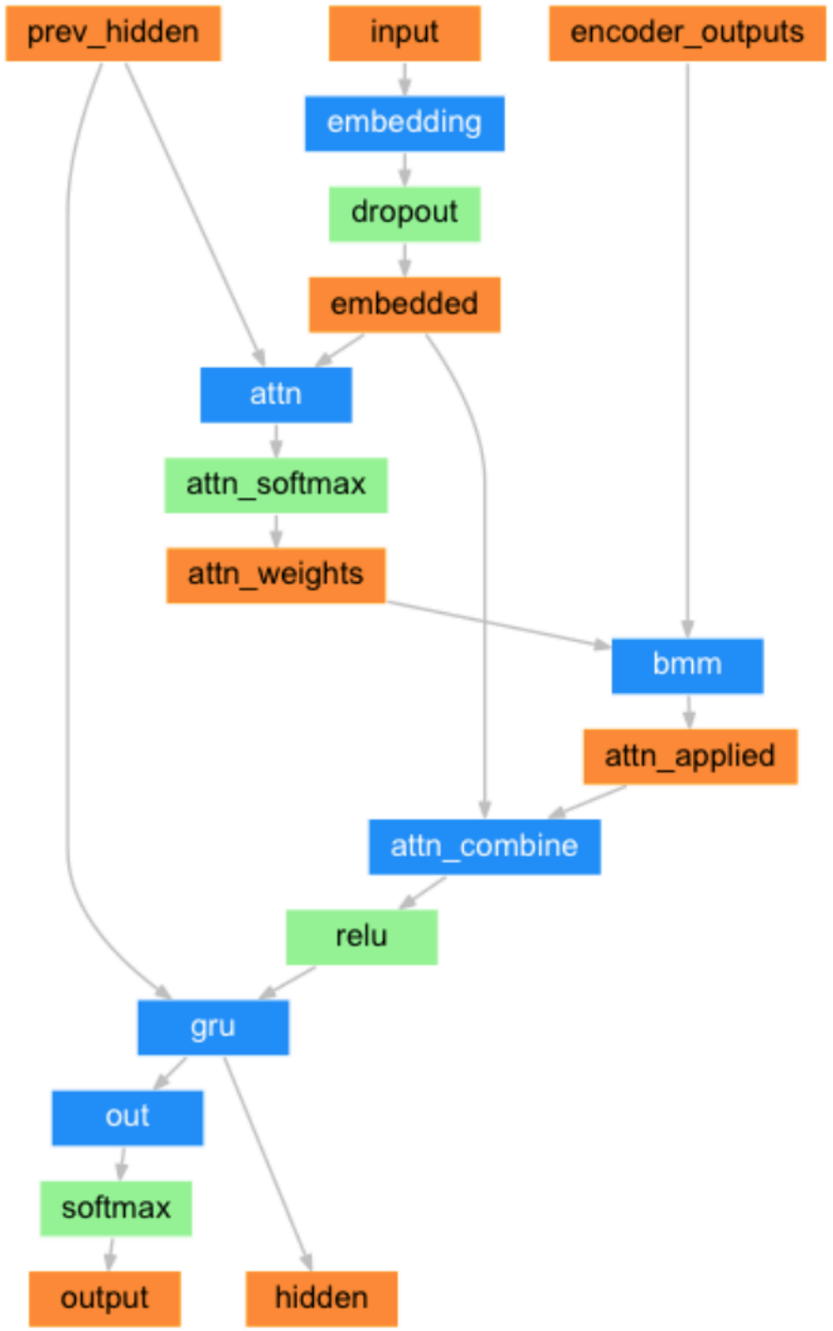
Simple Decoder

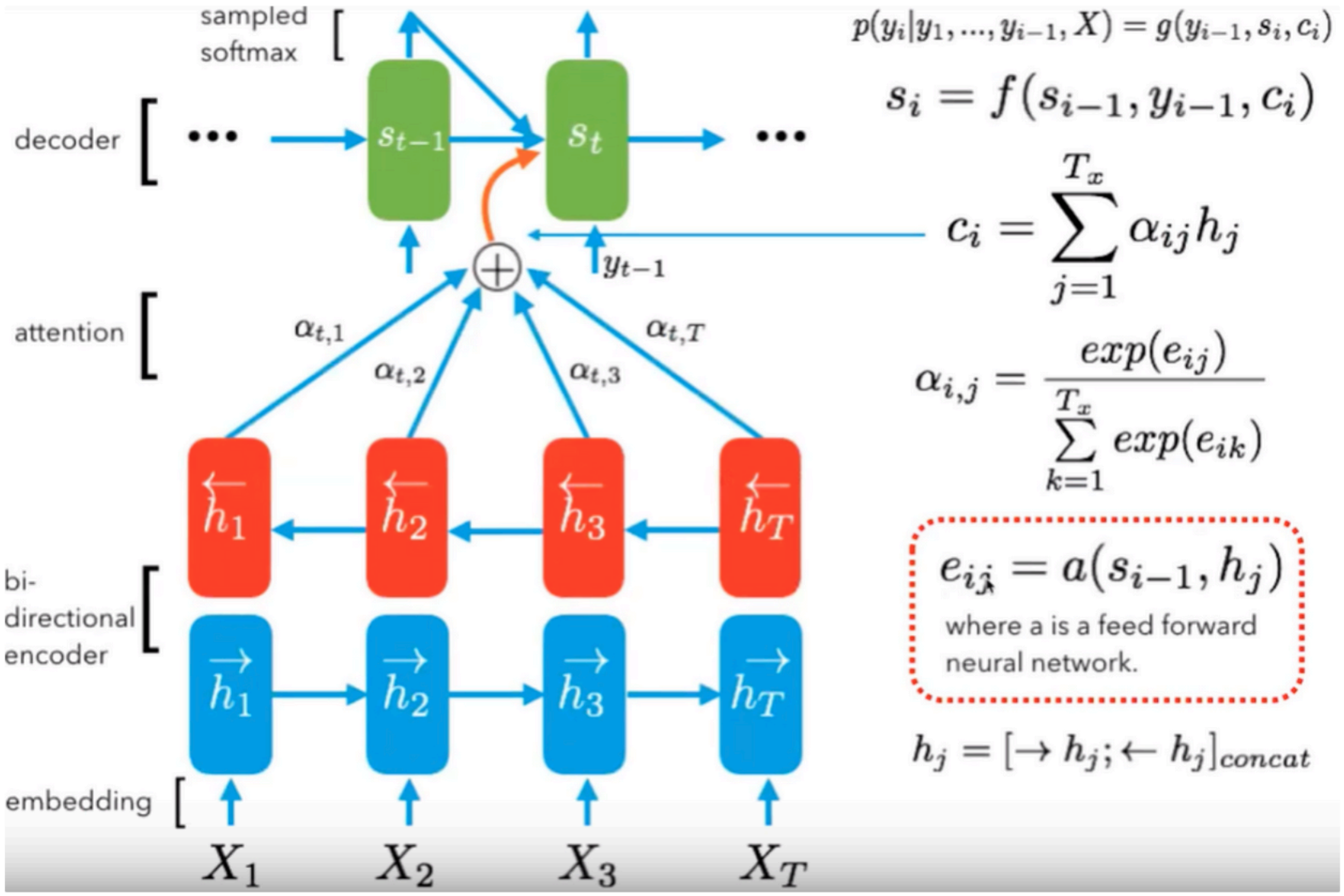



```
class DecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size):
        super(DecoderRNN, self).__init__()
        self.hidden_size = hidden_size

        self.embedding = nn.Embedding(output_size, hidden_size)
        self.gru = nn.GRU(hidden_size, hidden_size)
        self.out = nn.Linear(hidden_size, output_size)
        self.softmax = nn.LogSoftmax(dim=1)

    def forward(self, input, hidden):
        output = self.embedding(input).view(1, 1, -1)
        output = F.relu(output)
        output, hidden = self.gru(output, hidden)
        output = self.softmax(self.out(output[0]))
        return output, hidden

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

- 초기 input 토큰: start-of-string <SOS>
- 초기 hidden state: context vector (encoder의 마지막 hidden state)

Attention Decoder

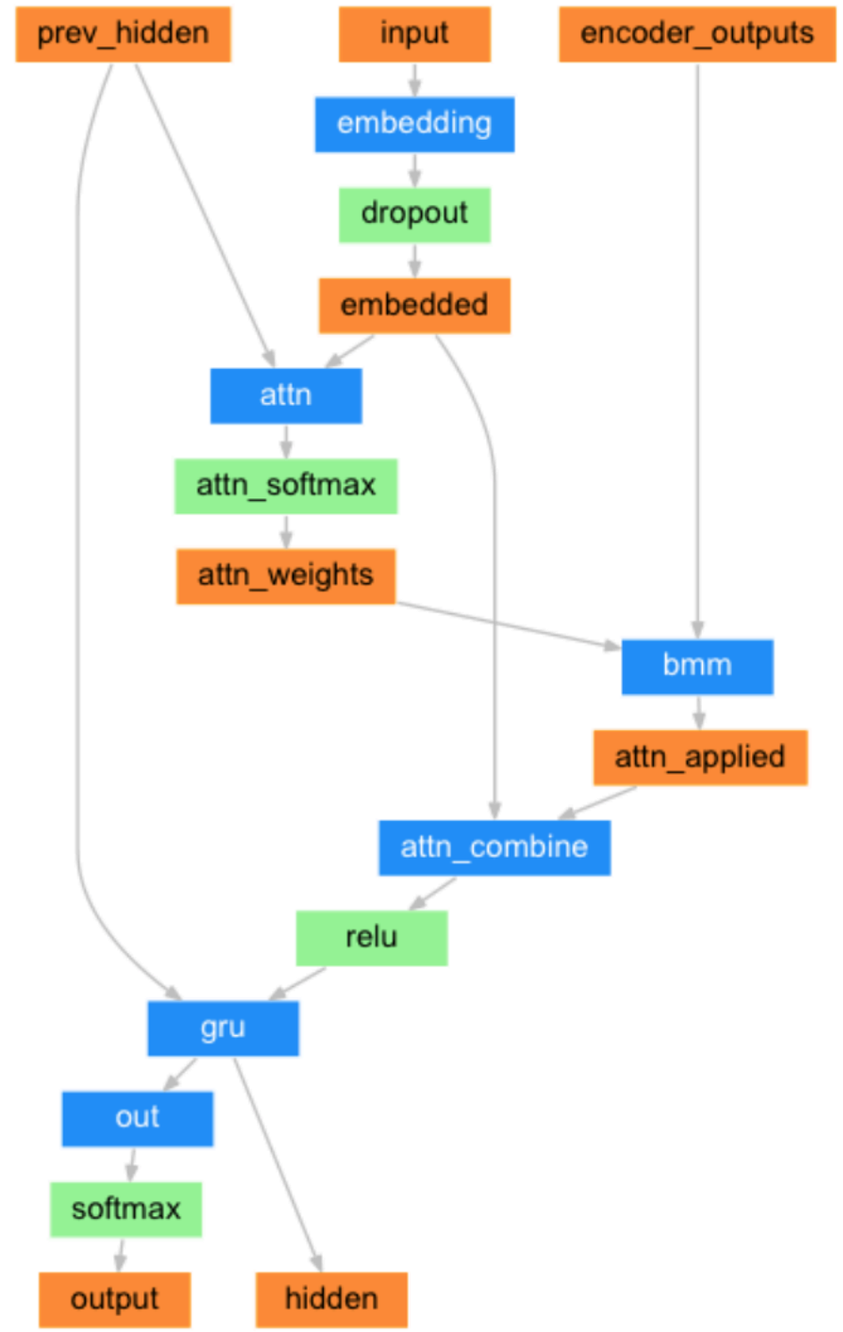




출처 : https://www.youtube.com/watch?v=upskBSbA9cA

3. Seq to seq 모델

Attention Decoder




```
class AttnDecoderRNN(nn.Module):
    def __init__(self, hidden_size, output_size, dropout_p=0.1, max_length=MAX_LENGTH):
        super(AttnDecoderRNN, self).__init__()
        self.hidden_size = hidden_size
        self.output_size = output_size
        self.dropout_p = dropout_p
        self.max_length = max_length

        self.embedding = nn.Embedding(self.output_size, self.hidden_size)
        self.attn = nn.Linear(self.hidden_size * 2, self.max_length)
        self.attn_combine = nn.Linear(self.hidden_size * 2, self.hidden_size)
        self.dropout = nn.Dropout(self.dropout_p)
        self.gru = nn.GRU(self.hidden_size, self.hidden_size)
        self.out = nn.Linear(self.hidden_size, self.output_size)

    def forward(self, input, hidden, encoder_outputs):
        embedded = self.embedding(input).view(1, 1, -1)
        embedded = self.dropout(embedded)

        attn_weights = F.softmax(
            self.attn(torch.cat((embedded[0], hidden[0]), 1)), dim=1)
        attn_applied = torch.bmm(attn_weights.unsqueeze(0),
                                 encoder_outputs.unsqueeze(0))

        output = torch.cat((embedded[0], attn_applied[0]), 1)
        output = self.attn_combine(output).unsqueeze(0)

        output = F.relu(output)
        output, hidden = self.gru(output, hidden)

        output = F.log_softmax(self.out(output[0]), dim=1)
        return output, hidden, attn_weights

    def initHidden(self):
        return torch.zeros(1, 1, self.hidden_size, device=device)
```

4. 학습

```
def indexesFromSentence(lang, sentence):
    return [lang.word2index[word] for word in sentence.split(' ')]


def tensorFromSentence(lang, sentence):
    indexes = indexesFromSentence(lang, sentence)
    indexes.append(EOS_token)
    return torch.tensor(indexes, dtype=torch.long, device=device).view(-1, 1)
```

```
def tensorsFromPair(pair):
    input_tensor = tensorFromSentence(input_lang, pair[0])
    target_tensor = tensorFromSentence(output_lang, pair[1])
    return (input_tensor, target_tensor)
```

- 학습 데이터 준비:
  - Pair sentences (영어, 프랑스어)로부터 input, target tensor 만들기
  - 끝에는 <EOS> 추가

4. 학습

```
def indexesFromSentence(lang, sentence):
    return [lang.word2index[word] for word in sentence.split(' ')]


def tensorFromSentence(lang, sentence):
    indexes = indexesFromSentence(lang, sentence)
    indexes.append(EOS_token)
    return torch.tensor(indexes, dtype=torch.long, device=device).view(-1, 1)
```

```
def tensorsFromPair(pair):
    input_tensor = tensorFromSentence(input_lang, pair[0])
    target_tensor = tensorFromSentence(output_lang, pair[1])
    return (input_tensor, target_tensor)
```

- 학습 데이터 준비:
  - Pair sentences (영어, 프랑스어)로부터 input, target tensor 만들기
  - 끝에는 <EOS> 추가

4. 학습

```
teacher_forcing_ratio = 0.5


def train(input_tensor, target_tensor, encoder, decoder, encoder_optimizer,
decoder_optimizer, criterion, max_length=MAX_LENGTH):
    encoder_hidden = encoder.initHidden()

    encoder_optimizer.zero_grad()
    decoder_optimizer.zero_grad()

    input_length = input_tensor.size(0)
    target_length = target_tensor.size(0)

    encoder_outputs = torch.zeros(max_length, encoder.hidden_size, device=device)

    loss = 0

    for ei in range(input_length):
        encoder_output, encoder_hidden = encoder(
            input_tensor[ei], encoder_hidden)
        encoder_outputs[ei] = encoder_output[0, 0]

    decoder_input = torch.tensor([[SOS_token]], device=device)

    decoder_hidden = encoder_hidden

    use_teacher_forcing = True if random.random() < teacher_forcing_ratio else False

    if use_teacher_forcing:
        # Teacher forcing: Feed the target as the next input
        for di in range(target_length):
            decoder_output, decoder_hidden, decoder_attention = decoder(
                decoder_input, decoder_hidden, encoder_outputs)
            loss += criterion(decoder_output, target_tensor[di])
            decoder_input = target_tensor[di]  # Teacher forcing
    else:
        # Without teacher forcing: use its own predictions as the next input
        for di in range(target_length):
            decoder_output, decoder_hidden, decoder_attention = decoder(
                decoder_input, decoder_hidden, encoder_outputs)
            topv, topi = decoder_output.topk(1)
            decoder_input = topi.squeeze().detach()  # detach from history as input

            loss += criterion(decoder_output, target_tensor[di])
            if decoder_input.item() == EOS_token:
                break

    loss.backward()

    encoder_optimizer.step()
    decoder_optimizer.step()

    return loss.item() / target_length
```

- Teacher forcing:
- Decoder의 다음 input으로 real target output을 사용하는 방법

4. 학습

```
def trainIters(encoder, decoder, n_iters, print_every=1000, plot_every=100,
learning_rate=0.01):
    start = time.time()
    plot_losses = []
    print_loss_total = 0  # Reset every print_every
    plot_loss_total = 0  # Reset every plot_every

    encoder_optimizer = optim.SGD(encoder.parameters(), lr=learning_rate)
    decoder_optimizer = optim.SGD(decoder.parameters(), lr=learning_rate)
    training_pairs = [tensorsFromPair(random.choice(pairs))
                      for i in range(n_iters)]
    criterion = nn.NLLLoss()

    for iter in range(1, n_iters + 1):
        training_pair = training_pairs[iter - 1]
        input_tensor = training_pair[0]
        target_tensor = training_pair[1]

        loss = train(input_tensor, target_tensor, encoder,
                     decoder, encoder_optimizer, decoder_optimizer, criterion)
        print_loss_total += loss
        plot_loss_total += loss

        if iter % print_every == 0:
            print_loss_avg = print_loss_total / print_every
            print_loss_total = 0
            print('%s (%d %d%%) %.4f' % (timeSince(start, iter / n_iters),
                                         iter, iter / n_iters * 100, print_loss_avg))

        if iter % plot_every == 0:
            plot_loss_avg = plot_loss_total / plot_every
            plot_losses.append(plot_loss_avg)
            plot_loss_total = 0

    showPlot(plot_losses)
```

- 초기화: optimizer, criterion
- 학습 iteration 시작
  - 학습 데이터 pair 생성
  - Loss 함수 그리기

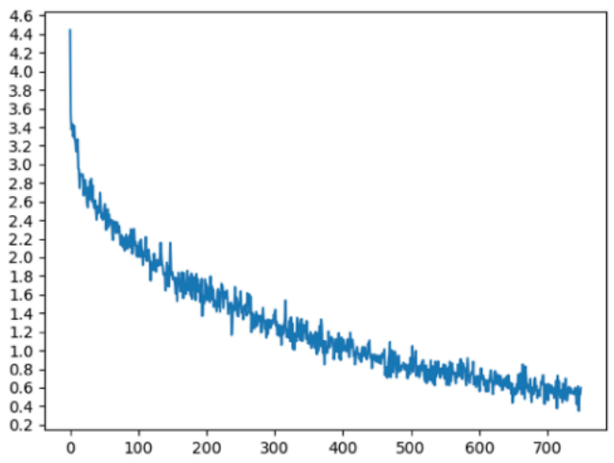
5. 평가

```
def evaluate(encoder, decoder, sentence, max_length=MAX_LENGTH):
    with torch.no_grad():
        input_tensor = tensorFromSentence(input_lang, sentence)
        input_length = input_tensor.size()[0]
        encoder_hidden = encoder.initHidden()

        encoder_outputs = torch.zeros(max_length, encoder.hidden_size, device=device)

        for ei in range(input_length):
            encoder_output, encoder_hidden = encoder(input_tensor[ei],
                                                     encoder_hidden)
            encoder_outputs[ei] += encoder_output[0, 0]

        decoder_input = torch.tensor([[SOS_token]], device=device)  # SOS

        decoder_hidden = encoder_hidden

        decoded_words = []
        decoder_attentions = torch.zeros(max_length, max_length)

        for di in range(max_length):
            decoder_output, decoder_hidden, decoder_attention = decoder(
                decoder_input, decoder_hidden, encoder_outputs)
            decoder_attentions[di] = decoder_attention.data
            topv, topi = decoder_output.data.topk(1)
            if topi.item() == EOS_token:
                decoded_words.append('<EOS>')
                break
            else:
                decoded_words.append(output_lang.index2word[topi.item()])

            decoder_input = topi.squeeze().detach()

        return decoded_words, decoder_attentions[:di + 1]
```

- Target이 없음
- Attention output 저장 (display)
- <EOS> 토큰에서 중지

6. 결과

```
hidden_size = 256
encoder1 = EncoderRNN(input_lang.n_words, hidden_size).to(device)
attn_decoder1 = AttnDecoderRNN(hidden_size, output_lang.n_words, dropout_p=0.1).to(device)

trainIters(encoder1, attn_decoder1, 75000, print_every=5000)
```



6. 결과

튜토리얼

```
evaluateRandomly(encoder1, attn_decoder1)
```

Out:

```
> j en suis contente .
= i m happy with that .
< i m happy with it . <EOS>

> il est impossible de le battre .
= he is impossible to beat .
< he is impossible to beat . <EOS>

> ils se trouvent juste derriere moi .
= they re right behind me .
< they re right behind me . <EOS>

> vous etes en avance .
= you are early .
< you are early . <EOS>

> il n est pas tout a fait la .
= he s not all there .
< he s not all there . <EOS>
```

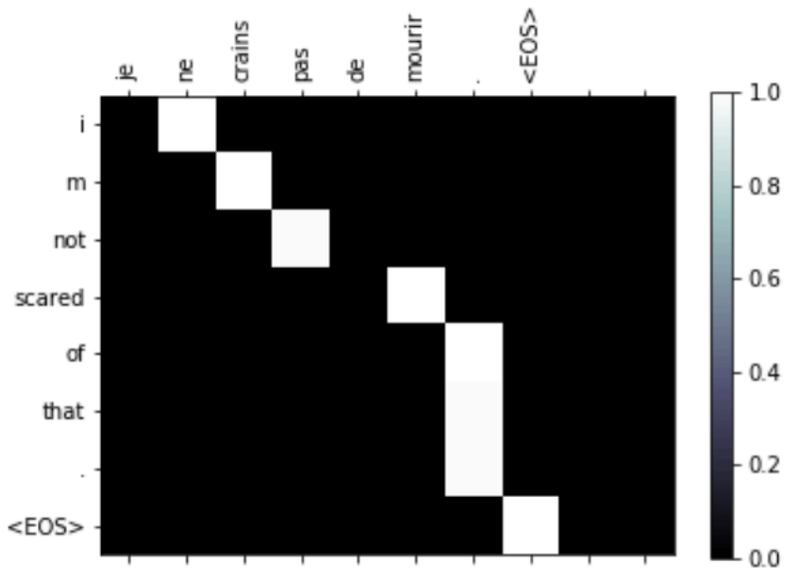
My Run

```
evaluateRandomly(encoder1, attn_decoder1)
```

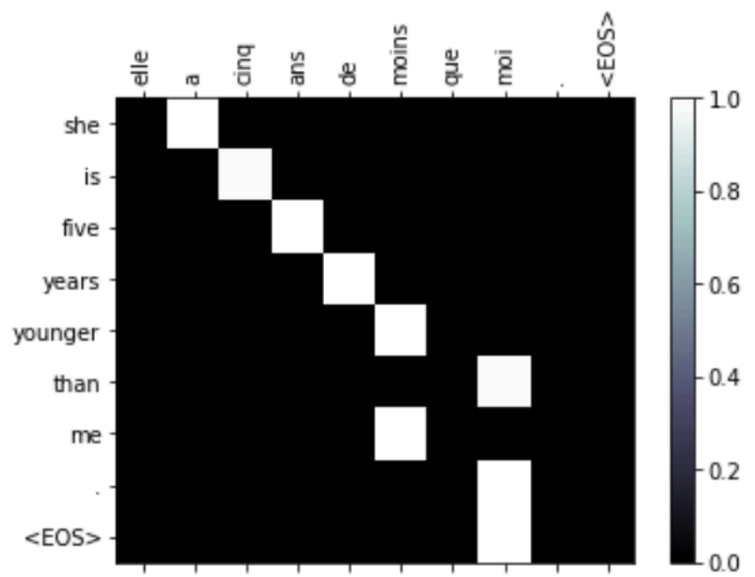
```
> je suis sur que tout ira bien .
= i m sure everything will be fine .
< i m sure everything will be fine . <EOS>

> tu me mens .
= you re lying to me .
< you re lying to me . <EOS>

> je suis tres heureux de vous voir .
= i am very glad to see you .
< i m very glad to see you you you .

> je suis creve .
= i am exhausted .
< i m exhausted . <EOS>

> j ai le meme age .
= i am the same age .
< i am the same age . <EOS>

> me voila satisfaite .
= i m contented .
< i m careful . <EOS>

> jamais il n a ete amoureux auparavant .
= he s never been in love before .
< he s never been in in love . <EOS>

> nous y sommes tous ensemble .
= we re all in this together .
< we re all together together . <EOS>

> on va jouer au tennis .
= we re going to play tennis .
< we re going to play tennis . <EOS>

> j ai soif .
= i m thirsty .
< i am thirsty . <EOS>
```

6. 결과

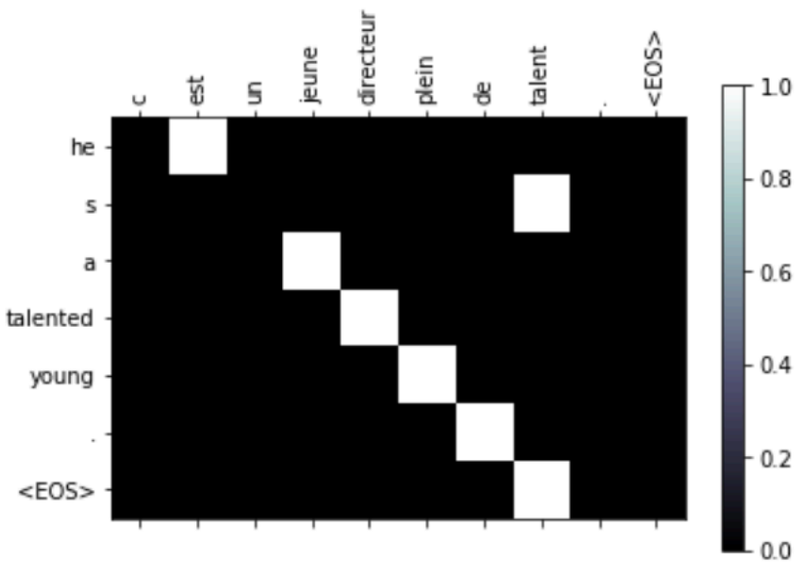
```
input = je ne crains pas de mourir .
output = i m not scared of that . <EOS>
```




```
input = elle a cinq ans de moins que moi .
output = she is five years younger than me . <EOS>
```




```
input = c est un jeune directeur plein de talent .
output = he s a talented young . <EOS>
```




```
input = elle est trop petit .
output = she s too slow . <EOS>
```