Introduction to

Seq2Seq Model

for Eng-Kor Translation

00 Outline

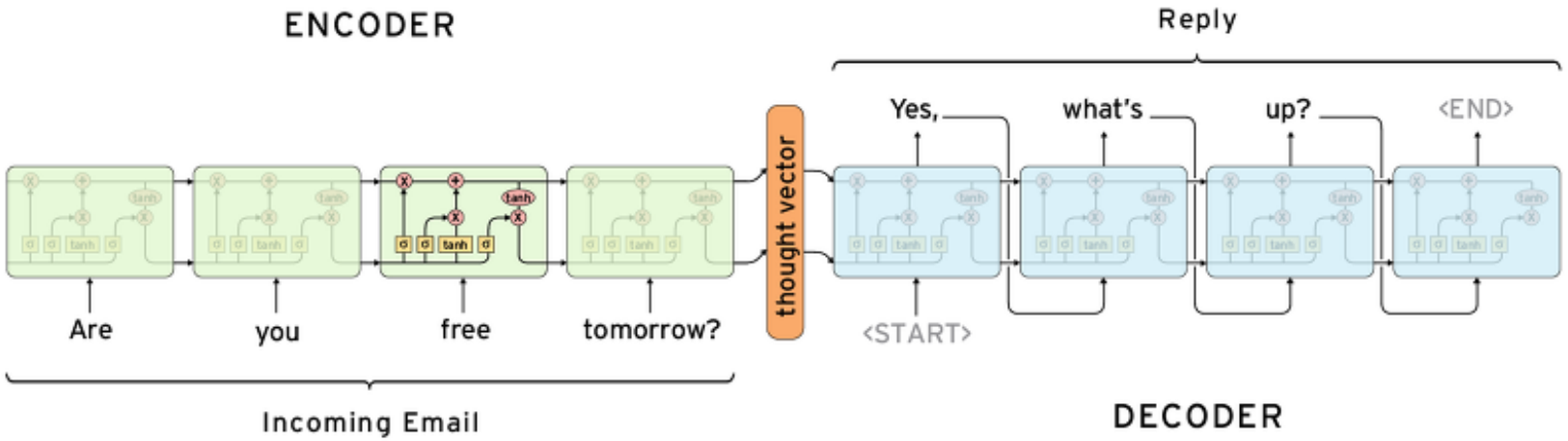
- Basic Concept
- Seq2Seq Vanilla
- Seq2Seq Advanced
- Seq2Seq Applying

Seq2Seq Concept

... transfer a sequence to another sequence ....

01 Concept

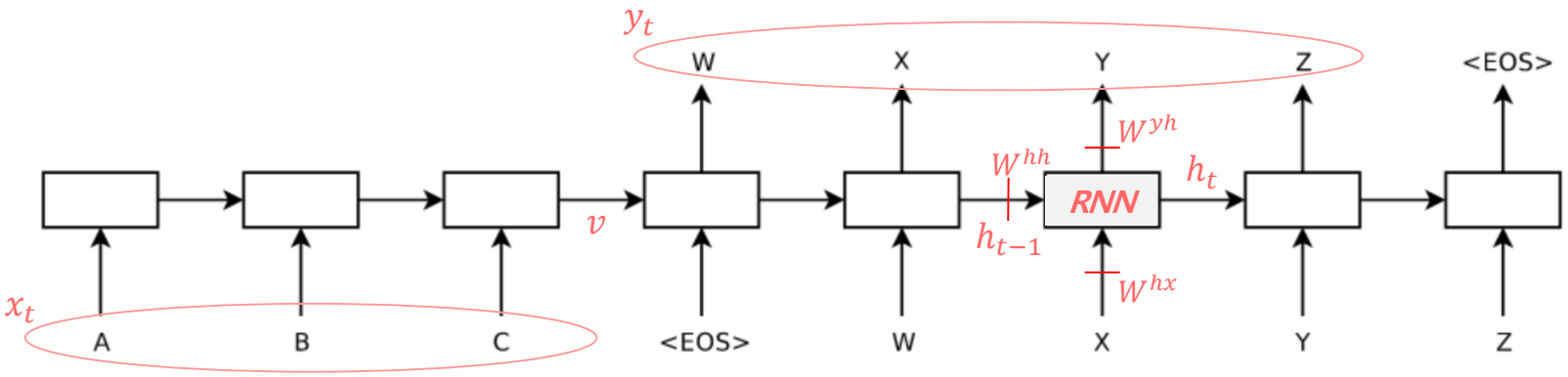
특정 도메인의 시퀀스를 입력으로 받아 다른 도메인의 시퀀스를 출력



[적용 사례]

- **챗봇**: 입력 시퀀스= 질문, 출력 시퀀스 = 답변
- **기계번역**: 입력 = 원어 문장, 출력 = 번역 문장
- **내용요약**: 입력 = 원문, 출력 = 요약문
- **STT**: 입력 = 음성언어, 출력 = 인쇄언어

Mechanism



$$h_t = \text{sigm}\left(W^{\text{hx}}x_t + W^{\text{hh}}h_{t-1}\right)$$
$$y_t = W^{\text{yh}}h_t$$

- W^{hx} : Weights between hidden node h and input node x
- W^{hh} : Weights between h_t and h_{t-1}

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

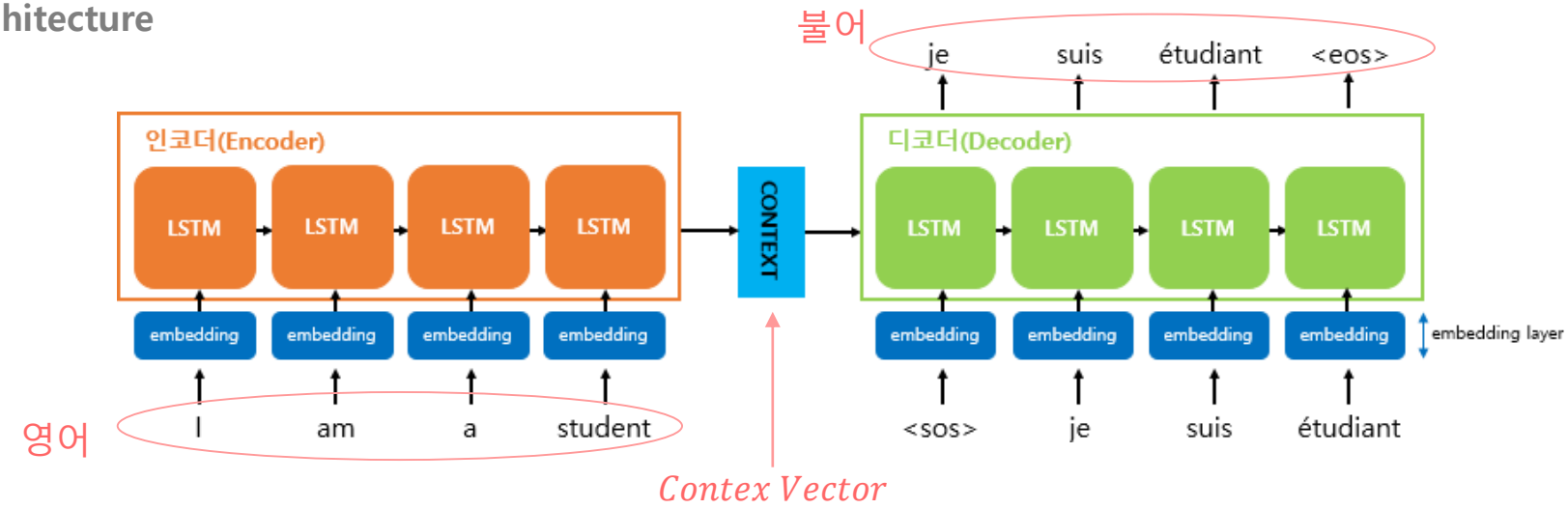
- T 길이의 Input Sentence가 주어질때 T' 길이의 Output을 생성
- Encoder에서 나온 Cell State인 v와 y_1, ..., y_t-1까지의 Input이 주어졌을때 y_t일 확률을 모두 곱하는 방식
- 가장 확률(Softmax)이 높은 y_t가 Output으로 채택됨

Char-level Translator

... translate english to french and korean ....

03 Char-level Translator – English-French

Architecture



[Embedded Vector]

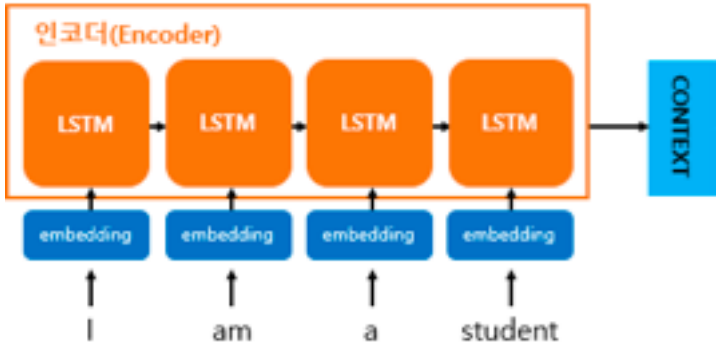
I	am	a	student
0.157	0.78	0.75	0.88
-0.25	0.29	-0.81	-0.17
0.478	-0.96	0.96	0.29
-0.78	0.52	0.12	0.48

- 4차원 크기의 임베딩 벡터를 예로 들어 설명
- 보통 수백 차원 이상 필요

04 Char-level Translator – English-French

Encoder

- 입력 문장을 단어 (또는 글자) 단위로 토큰화
- 각 토큰은 LSTM Cell의 각 시점에서의 입력값
- <eos> 토큰이 입력되면 마지막 시점의 hidden state를 디코더로 전달 (Context Vector)

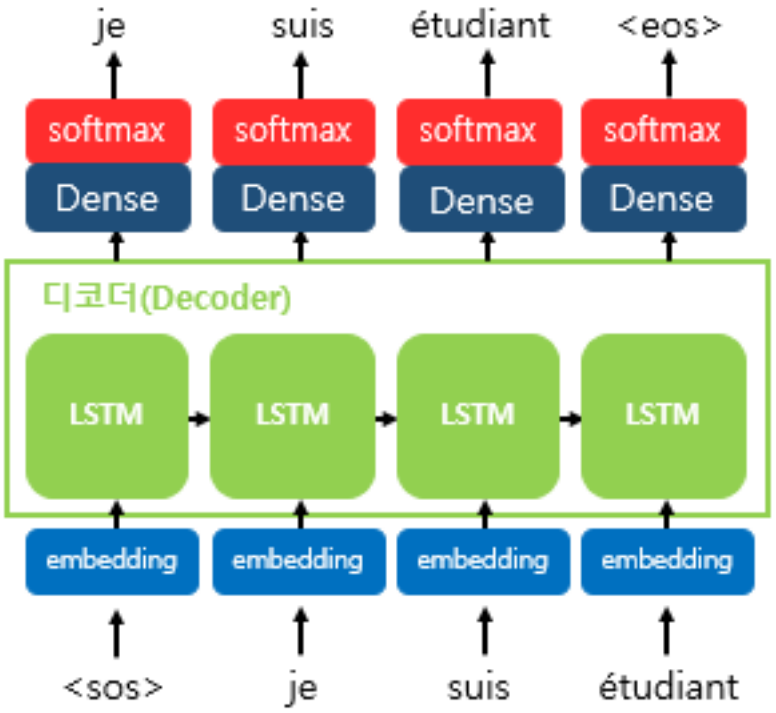


인코더에서는 출력값 를 사용하지 않음

05 Char-level Translator – English-French

Decoder

- 최초 입력 토큰 - <sos>
- <sos>와 Context Vector를 사용하여 등장 확률이 높은 단어(또는 글자) y_t를 산출
- <eos> 입력시 Prediction 종료



Softmax 함수를 통해 출력 시퀀스의 각 단어별 확률값 계산 및 출력 단어 결정

05 Char-level Translator – English-French

Original Data

- http://www.manythings.org/anki → fra-eng.zip → fra.txt

```
lines= pd.read_csv('fra.txt', names=['src', 'tar', 'Comment'], sep='\t')
```

	src	tar	Comment
47190	I considered not going.	J'ai songé à ne pas y aller.	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
40666	How much is this ring?	Combien coûte cette bague ?	CC-BY 2.0 (France) Attribution: tatoeba.org #4...
52589	Did you write this book?	Tu as écrit ce livre ?	CC-BY 2.0 (France) Attribution: tatoeba.org #8...
24793	It has no parallel.	Ça n'a pas de parallèle.	CC-BY 2.0 (France) Attribution: tatoeba.org #4...
8984	Thanks a bunch.	Mille mercis.	CC-BY 2.0 (France) Attribution: tatoeba.org #4...
30605	No one believed him.	Personne ne l'a cru.	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
47995	I think about it often.	J'y pense souvent.	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
31776	Tom likes traveling.	Tom aime voyager.	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
15855	Prepare yourself.	Prépare-toi !	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
21271	Tom wants revenge.	Tom veut se venger.	CC-BY 2.0 (France) Attribution: tatoeba.org #2...

05 Char-level Translator – English-French

Character based tokenization

- 예제 프로그램을 간단히 구성하기 위해 토큰의 단위가 Word가 아닌 Character

```
src_vocab=set()
for line in lines.src:      # 1줄씩 읽음
    for char in line:       # 1글자씩 읽음
        src_vocab.add(char)
```

```
['W', 'X', 'Y', 'Z', 'a', 'b', 'c', 'd', 'e', 'f',
'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p',
'q', 'r', 's', 't', 'u', 'v', 'w', 'x', 'y', 'z']
```

```
tar_vocab=set()
for line in lines.tar:      # 1줄씩 읽음
    for char in line:       # 1글자씩 읽음
        tar_vocab.add(char)
```

```
['T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a', 'b', 'c',
'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm',
'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v', 'w']
```

05 Char-level Translator – English-French

Indexing

- Character 정렬 후 인덱스 부여 → Dictionary 구축 ((char, index) 쌍)

```
src_to_index = dict([(word, i+1) for i, word in enumerate(src_vocab)])
tar_to_index = dict([(word, i+1) for i, word in enumerate(tar_vocab)])
```

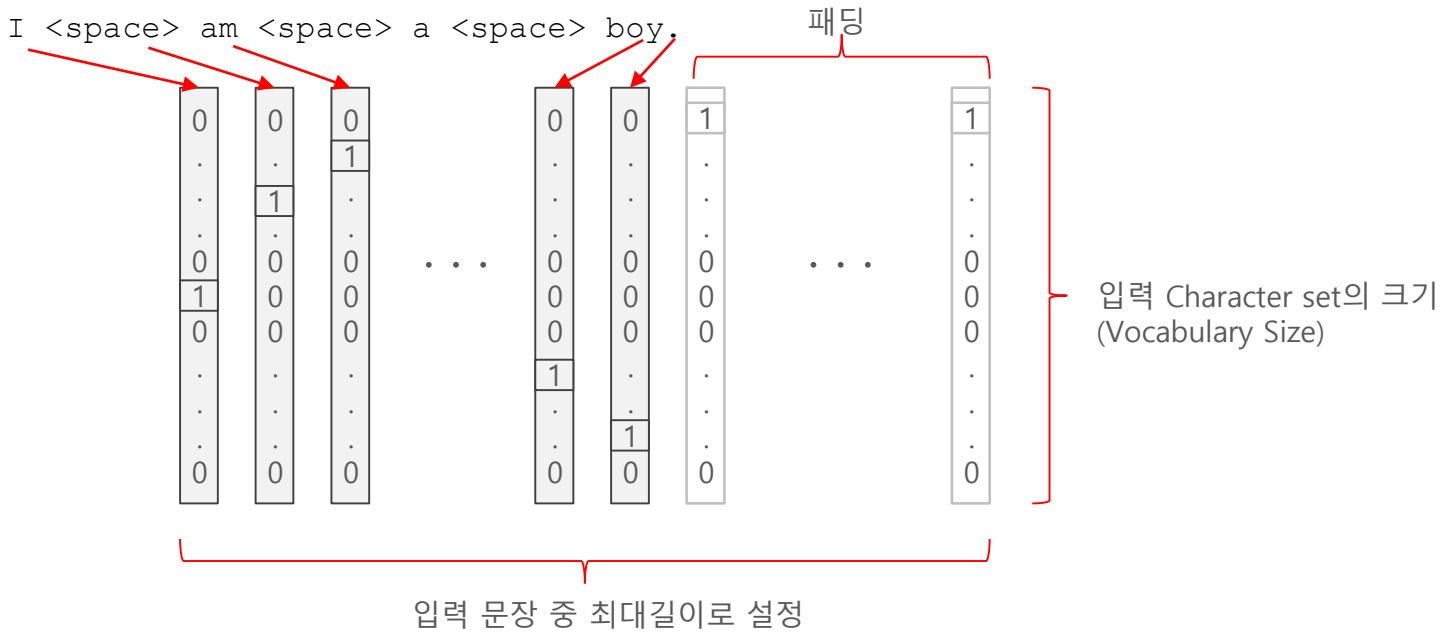
```
{' ': 1, '!': 2, '"': 3, '$': 4, '%': 5, '&': 6, "'": 7, ',': 8, '-': 9, '.': 10, '/': 11, '0': 12, '1': 13, '2': 14, '3': 15, '4': 16,
'5': 17, '6': 18, '7': 19, '8': 20, '9': 21, ':': 22, '?': 23, 'A': 24, 'B': 25, 'C': 26, 'D': 27, 'E': 28, 'F': 29, 'G': 30, 'H': 31, 'I':
32, 'J': 33, 'K': 34, 'L': 35, 'M': 36, 'N': 37, 'O': 38, 'P': 39, 'Q': 40, 'R': 41, 'S': 42, 'T': 43, 'U': 44, 'V': 45, 'W': 46, 'X': 47,
'Y': 48, 'Z': 49, 'a': 50, 'b': 51, 'c': 52, 'd': 53, 'e': 54, 'f': 55, 'g': 56, 'h': 57, 'i': 58, 'j': 59, 'k': 60, 'l': 61, 'm': 62, 'n':
63, 'o': 64, 'p': 65, 'q': 66, 'r': 67, 's': 68, 't': 69, 'u': 70, 'v': 71, 'w': 72, 'x': 73, 'y': 74, 'z': 75, 'é': 76, '’': 77, '€': 78}
{'\t': 1, '\n': 2, ' ': 3, '!': 4, '"': 5, '$': 6, '%': 7, '&': 8, "'": 9, '(': 10, ')': 11, ',': 12, '-': 13, '.': 14, '0': 15, '1': 16,
'2': 17, '3': 18, '4': 19, '5': 20, '6': 21, '7': 22, '8': 23, '9': 24, ':': 25, '?': 26, 'A': 27, 'B': 28, 'C': 29, 'D': 30, 'E': 31, 'F':
32, 'G': 33, 'H': 34, 'I': 35, 'J': 36, 'K': 37, 'L': 38, 'M': 39, 'N': 40, 'O': 41, 'P': 42, 'Q': 43, 'R': 44, 'S': 45, 'T': 46, 'U': 47,
'V': 48, 'W': 49, 'X': 50, 'Y': 51, 'Z': 52, 'a': 53, 'b': 54, 'c': 55, 'd': 56, 'e': 57, 'f': 58, 'g': 59, 'h': 60, 'i': 61, 'j': 62, 'k':
63, 'l': 64, 'm': 65, 'n': 66, 'o': 67, 'p': 68, 'q': 69, 'r': 70, 's': 71, 't': 72, 'u': 73, 'v': 74, 'w': 75, 'x': 76, 'y': 77, 'z': 78,
'\xa0': 79, '«': 80, '»': 81, 'À': 82, 'Ç': 83, 'É': 84, 'Ê': 85, 'Ô': 86, 'à': 87, 'â': 88, 'ç': 89, 'è': 90, 'é': 91, 'ê': 92, 'ë': 93,
'î': 94, 'ï': 95, 'ô': 96, 'ù': 97, 'û': 98, 'œ': 99, 'C': 100, '\u2009': 101, '\u200b': 102, '‘': 103, '’': 104, '\u202f': 105}
```

Char-level Translator – English-French

One-hot encoding

- 예측 값과의 오차 측정에 사용되는 실제값 뿐아니라 입력값도 One-hot vector 적용

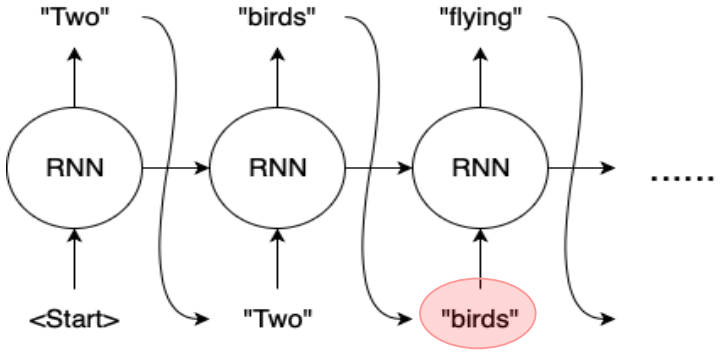
```
from tensorflow.keras.utils import to_categorical
encoder_input = to_categorical(encoder_input)
decoder_input = to_categorical(decoder_input)
decoder_target = to_categorical(decoder_target)
```



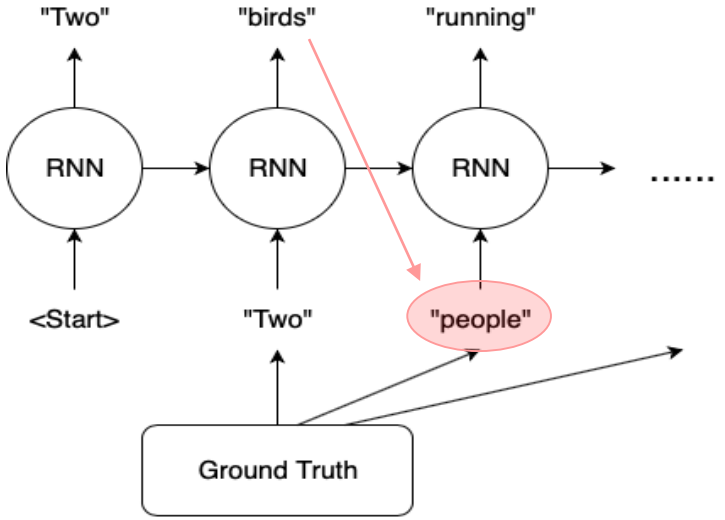
Char-level Translator – English-French

Teacher Forcing

- 훈련 과정에서는 이전 시점의 디코더 셀의 출력을 현재 시점의 디코더 셀의 입력으로 넣어주지 않고, 이전 시점의 실제값을 현재 시점의 디코더 셀의 입력값으로 사용
- 이전 시점의 디코더 셀의 예측이 틀렸는데 이를 현재 시점의 디코더 셀의 입력으로 사용하면 현재 시점의 디코더 셀의 예측도 잘 못될 가능성이 높고 연쇄적으로 디코더 전체의 예측 성능 저하



Without Teacher Forcing



With Teacher Forcing

08 Char-level Translator – English-French

Train Model Design

인코더

```
from tensorflow.keras.layers import Input, LSTM, Embedding, Dense
from tensorflow.keras.models import Model
encoder_inputs = Input(shape=(None, src_vocab_size))
encoder_lstm = LSTM(units=256, return_state=True)
encoder_outputs, state_h, state_c = encoder_lstm(encoder_inputs)
# encoder_outputs도 같이 리턴받기는 했지만 여기서는 필요없으므로 이 값은 버림
encoder_states = [state_h, state_c]
# LSTM은 바닐라 RNN과는 달리 상태가 두 개, 은닉 상태와 셀 상태
```

디코더

```
decoder_inputs = Input(shape=(None, tar_vocab_size))
decoder_lstm = LSTM(units=256, return_sequences=True, return_state=True)
decoder_outputs, _, _ = decoder_lstm(decoder_inputs, initial_state=encoder_states)
# 디코더의 첫 상태를 인코더의 은닉 상태, 셀 상태로 설정
decoder_softmax_layer = Dense(tar_vocab_size, activation='softmax')
decoder_outputs = decoder_softmax_layer(decoder_outputs)

model = Model([encoder_inputs, decoder_inputs], decoder_outputs)
model.compile(optimizer="rmsprop", loss="categorical_crossentropy")
```

09 Char-level Translator – English-French

Model Fitting

```
model.fit(x=[encoder_input, decoder_input], y=decoder_target, batch_size=64,
epochs=50, validation_split=0.2)
```

```
Epoch 38/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.1018 - val_loss: 0.5096
Epoch 39/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.1017 - val_loss: 0.5117
Epoch 40/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.1011 - val_loss: 0.5123
Epoch 41/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.1005 - val_loss: 0.5140
Epoch 42/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.1002 - val_loss: 0.5150
Epoch 43/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0995 - val_loss: 0.5159
Epoch 44/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0993 - val_loss: 0.5179
Epoch 45/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0989 - val_loss: 0.5184
Epoch 46/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0983 - val_loss: 0.5205
Epoch 47/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0979 - val_loss: 0.5230
Epoch 48/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0975 - val_loss: 0.5241
Epoch 49/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0972 - val_loss: 0.5254
Epoch 50/50
48000/48000 [==============================] - 109s 2ms/sample - loss: 0.0969 - val_loss: 0.5273
```

10 Char-level Translator – English-French

Prediction Model Design

```
encoder_model = Model(inputs=encoder_inputs, outputs=encoder_states)

decoder_state_input_h = Input(shape=(256,))
decoder_state_input_c = Input(shape=(256,))
decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]
decoder_outputs, state_h, state_c = decoder_lstm(decoder_inputs,
initial_state=decoder_states_inputs)
# 문장의 다음 단어를 예측하기 위해서 초기 상태를 이전 상태로 사용
decoder_states = [state_h, state_c]
# 이번에는 훈련 과정에서와 달리 은닉 상태와 셀 상태인 state_h와 state_c를 버리지 않음.
decoder_outputs = decoder_softmax_layer(decoder_outputs)
decoder_model = Model(inputs=[decoder_inputs] + decoder_states_inputs,
                      outputs=[decoder_outputs] + decoder_states)
```

11 Char-level Translator – English-French

Decode Sequence

```
def decode_sequence(input_seq):    # 입력으로부터 인코더의 상태를 얻음
    states_value = encoder_model.predict(input_seq)    # <SOS>에 해당하는 원-핫 벡터 생성
    target_seq = np.zeros((1, 1, tar_vocab_size))
    target_seq[0, 0, tar_to_index['\t']] = 1.
    stop_condition = False
    decoded_sentence = ""
    while not stop_condition: #stop_condition이 True가 될 때까지 루프 반복
        output_tokens, h, c = decoder_model.predict([target_seq] + states_value)
        sampled_token_index = np.argmax(output_tokens[0, -1, :])
        sampled_char = index_to_tar[sampled_token_index]
        decoded_sentence += sampled_char
        # <sos>에 도달하거나 최대 길이를 넘으면 중단
        if (sampled_char == '\n' or
           len(decoded_sentence) > max_tar_len):
            stop_condition = True        # 길이가 1인 타겟 시퀀스를 업데이트 합니다
        target_seq = np.zeros((1, 1, tar_vocab_size))
        target_seq[0, 0, sampled_token_index] = 1.        # 상태를 업데이트 합니다
        states_value = [h, c]
    return decoded_sentence
```

Char-level Translator – English-French

Result

```
import numpy as np

for seq_index in [3,50,100,300,1001]: # 입력 문장의 인덱스
    input_seq = encoder_input[seq_index: seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)
    print(35 * "-")
    print('입력 문장:', lines.src[seq_index])
    print('정답 문장:', lines.tar[seq_index][1:len(lines.tar[seq_index])-1])
    # '\t'와 '\n'을 빼고 출력
    print('번역기가 번역한 문장:', decoded_sentence[:len(decoded_sentence)-1])
    # '\n'을 빼고 출력
```

```
-----------------------------------
입력 문장: Run!
정답 문장:  Cours !
번역기가 번역한 문장:  Coursez-vous !
-----------------------------------
입력 문장: I lied.
정답 문장:  J'ai menti.
번역기가 번역한 문장:  J'ai menti.
-----------------------------------
입력 문장: Come in.
정답 문장:  Entre.
번역기가 번역한 문장:  Entre.
-----------------------------------
입력 문장: I did OK.
정답 문장:  Je m'en suis bien sortie.
번역기가 번역한 문장:  Je m'en suis bien sortie.
-----------------------------------
입력 문장: We're hot.
정답 문장:  Nous avons chaud.
번역기가 번역한 문장:  Nous avons attrapé.
-----------------------------------
입력 문장: Cheers!
정답 문장:  Tchin-tchin !
번역기가 번역한 문장:  À votre part.
-----------------------------------
입력 문장: Catch Tom.
정답 문장:  Attrape Tom.
번역기가 번역한 문장:  Attrapez Tom.
-----------------------------------
입력 문장: We're hot.
정답 문장:  Nous avons chaud.
번역기가 번역한 문장:  Nous avons attrapé.
-----------------------------------
입력 문장: We're sorry.
정답 문장:  Nous sommes désolées.
번역기가 번역한 문장:  Nous sommes désolés.
-----------------------------------
입력 문장: You idiot!
정답 문장:  Espèce d'idiot !
번역기가 번역한 문장:  Espèce d'imbécise !
```

13 Char-level Translator – English-Korean

Original Data

- http://www.manythings.org/anki → kor-eng.zip → kor.txt

```
lines= pd.read_csv('kor.txt', names=['src', 'tar', 'Comment'], sep='\t')
```

	src	tar	Comment
2154	Let's see if we can get inside.	우리가 안으로 들어갈 수 있는지 보자.	CC-BY 2.0 (France) Attribution: tatoeba.org #3...
666	There was blood.	피가 있었다.	CC-BY 2.0 (France) Attribution: tatoeba.org #4...
707	I like languages.	언어를 좋아합니다.	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
425	Keep singing.	계속 노래해.	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
957	I need to finish it.	나는 이걸 끝내야 해.	CC-BY 2.0 (France) Attribution: tatoeba.org #5...
1736	Tom was diagnosed with ASD.	톰은 자폐스펙트럼으로 진단받았어.	CC-BY 2.0 (France) Attribution: tatoeba.org #7...
390	Don't eat it.	먹지 마.	CC-BY 2.0 (France) Attribution: tatoeba.org #6...
1751	What's your Skype username?	네 스카이프 사용자 이름은 뭐야?	CC-BY 2.0 (France) Attribution: tatoeba.org #1...
2134	I don't want to be alone again.	난 더 이상 혼자이고 싶지 않아.	CC-BY 2.0 (France) Attribution: tatoeba.org #2...
345	Tom blushed.	톰의 얼굴이 빨개졌어.	CC-BY 2.0 (France) Attribution: tatoeba.org #1...

Char-level Translator – English-Korean

Result

```
import numpy as np

for seq_index in [3,50,100,300,1001]: # 입력 문장의 인덱스
    input_seq = encoder_input[seq_index: seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)
    print(35 * "-")
    print('입력 문장:', lines.src[seq_index])
    print('정답 문장:', lines.tar[seq_index][1:len(lines.tar[seq_index])-1])
    # '\t'와 '\n'을 빼고 출력
    print('번역기가 번역한 문장:', decoded_sentence[:len(decoded_sentence)-1])
    # '\n'을 빼고 출력
```

```
-----------------------------------
입력 문장: No way!
정답 문장:  절대 아니야.
번역기가 번역한 문장:  톰이 나이다.
-----------------------------------
입력 문장: I miss my cat.
정답 문장:  난 내 고양이가 그리워.
번역기가 번역한 문장:  나는 그것을 다고 싶어.
-----------------------------------
입력 문장: We were born to die.
정답 문장:  우리는 죽기 위해 태어났어.
번역기가 번역한 문장:  그 사람은 아직 일어야.
-----------------------------------
입력 문장: To tell the truth, he is not a human being.
정답 문장:  진실을 말하자면, 그는 인간이 아냐.
번역기가 번역한 문장:  이 사전은 정확히 하고 있어.
-----------------------------------
입력 문장: I now agree with Tom.
정답 문장:  나는 이제 톰의 의견에 동의한다.
번역기가 번역한 문장:  나는 톰이 아빠 거라고 생각해.
-----------------------------------
입력 문장: Oh please!
정답 문장:  아 제발!
번역기가 번역한 문장:  그 사람들은 이겼어.
-----------------------------------
입력 문장: Please listen.
정답 문장:  제발 좀 들어.
번역기가 번역한 문장:  톰이 웃었어.
-----------------------------------
입력 문장: I'm just doing my best.
정답 문장:  난 그저 최선을 다하고 있을 뿐이다.
번역기가 번역한 문장:  나는 톰이 아직도 프랑스어를 배워.
-----------------------------------
입력 문장: Why didn't you tell me you weren't going to be there?
정답 문장:  왜 네가 거기에 안 갈 거라고 말 나한테 안 했어?
번역기가 번역한 문장:  그 사람은 아직 도로 일해.
-----------------------------------
입력 문장: She majors in medicine.
정답 문장:  그녀는 의학을 전공하고 있습니다.
번역기가 번역한 문장:  그 사람은 오늘 밤에 있어.
```

Word-level Translator

... english to korean ....

15 Word-level Translator – English-Korean

Indexing

- Word 정렬 후 인덱스 부여 → Dictionary 구축 ((word, index) 쌍)

```
input_words = sorted(list(all_eng_words))
target_words = sorted(list(all_korean_words))

input_token_index = dict([(word, i) for i, word in enumerate(input_words)])
target_token_index = dict([(word, i) for i, word in enumerate(target_words)])
```

```
{'COMMA': 0, 'COMMA°c': 1, 'a': 2, 'able': 3, 'aboard': 4, 'about': 5, 'above': 6, 'abroad': 7, 'absolutely': 8, 'abusive': 9,
'accept': 10, 'accident': 11, 'accomplishments': 12, 'account': 13, 'accurate': 14, 'act': 15, 'acted': 16, 'acting': 17, 'act
inium': 18, 'action': 19, 'actions': 20, 'active': 21, 'actor': 22, 'acts': 23, 'actually': 24, 'add': 25, 'addict': 26, 'addi
cted': 27, 'addictive': 28, 'address': 29, 'admission': 30, 'adult': 31, 'advice': 32, 'afraid': 33, 'africa': 34, 'after': 3
5, 'afternoon': 36, 'again': 37, 'against': 38, 'age': 39, 'ages': 40, 'ago': 41, 'agree': 42, 'agreed': 43, 'ahead': 44, 'ai
r': 45, 'airport': 46, 'alarm': 47, 'alcoholic': 48, 'alice': 49, 'all': 50, 'allergic': 51, 'allow': 52, 'allowed': 53, 'almo
st': 54, 'alone': 55, 'along': 56, 'already': 57, 'also': 58, 'although': 59, 'always': 60, 'am': 61, 'amateurs': 62, 'amazin
g': 63, 'american': 64, 'ammo': 65, 'an': 66, 'ancient': 67, 'and': 68, 'anger': 69, 'angles': 70, 'angry': 71, 'animal': 72,
```

```
{'COMMA': 0, 'COMMA°c일': 1, 'START_': 2, '_END': 3, 'a와': 4, 'birthday': 5, 'b의': 6, 'dna': 7, 'd를': 8, 'happy': 9, 'mary
가': 10, 'tom과': 11, 'tom은': 12, 'tom이': 13, '가': 14, '가게': 15, '가격은': 16, '가격을': 17, '가격이': 18, '가고': 19, '가곤': 20,
'가기': 21, '가기로': 22, '가까운': 23, '가까이': 24, '가까이서': 25, '가끔': 26, '가난한': 27, '가난했었다': 28, '가난했었어': 29, '가는': 30,
'가는지': 31, '가능성이': 32, '가능하지': 33, '가능한': 34, '가능해': 35, '가도': 36, '가라고': 37, '가려': 38, '가르쳐': 39, '가르쳤는지': 40,
'가르쳤어': 41, '가르칠': 42, '가리는': 43, '가만히': 44, '가방': 45, '가방에': 46, '가방이': 47, '가버려서': 48, '가벼워': 49, '가본': 50, '가
봐야': 51, '가봤지': 52, '가상해': 53, '가서': 54, '가설이': 55, '가수가': 56, '가수는': 57, '가수야': 58, '가야해': 59, '가위': 60, '가을보
다': 61, '가을이': 62, '가자': 63, '가장': 64, '가정': 65, '가져': 66, '가져갈': 67, '가져도': 68, '가져와': 69, '가졌다': 70, '가족': 71,
'가족과': 72, '가족은': 73, '가지': 74, '가지고': 75, '가지세요': 76, '가진': 77, '가질': 78, '가짜같진': 79, '가치': 80, '가치가': 81, '가호
```

16 Word-level Translator – English-Korean

Embedding vector

- 원-핫 인코딩 (2만 차원 이상; Sparse vector) → Embedding vector (1,024차원 이하; Dense vector)

```
text = [['Hope', 'to', 'see', 'you', 'soon'], ['Nice', 'to', 'see, 'you', 'again']]
```

정수 인코딩

```
text = [[0, 1, 2, 3, 4],[5, 1, 2, 3, 6]]
```

임베딩 선언

```
from keras.layers import Embedding
Embedding(7, 2, input_length=5)   # 7:vocab size, 2:embedding vector size
```

임베딩 결과

Index	Embedding
0	[1.2, 3.1]
1	[0.1, 4.2]
2	[1.0, 3.1]
3	[0.3, 2.1]
4	[2.2, 1.4]
5	[0.7, 1.7]
6	[4.1, 2.0]

17 Word-level Translator – English-Korean

Result

```
for seq_index in [3077,2122,1035,1064, 256, 3068, . . . , 2136, 1150, 3153]:
    input_seq = encoder_input_data[seq_index: seq_index + 1]
    decoded_sentence = decode_sequence(input_seq)    #print('-')
    print(35 * "-")
    print('입력 문장:', lines.eng[seq_index])
    print('번역한 문장:', decoded_sentence[:len(decoded_sentence)-4])
    print('정답 문장:', lines.kor[seq_index][7:len(lines.kor[seq_index])-4])
```

```
-----------------------------------
입력 문장: tom had difficulty concentrating on his work
번역한 문장:  톰은 자기 분야의 껐어
정답 문장: 톰이 일에 집중하는 데 어려움을 겪더라
-----------------------------------
입력 문장: five times five is twenty five
번역한 문장:  네 딸은 너무 언어로 수 있어
정답 문장: 오 곱하기 오는 이십오야
-----------------------------------
입력 문장: i went home by train
번역한 문장:  나는 사람들은 이상 노력했어
정답 문장: 나는 기차로 집에 갔어
-----------------------------------
입력 문장: tom baked three pies
번역한 문장:  톰은 자기 분야의 껐어
정답 문장: 톰은 파이를 세개 구웠어
-----------------------------------
입력 문장: once again
번역한 문장:  제발
정답 문장: 한 번 더
-----------------------------------
입력 문장: ive read your book it was very interesting
번역한 문장:  나는 사람들은 이상 게 좋아
정답 문장: 당신 책을 읽어봤었어요 꽤 흥미롭더라구요
-----------------------------------
입력 문장: hes autistic
번역한 문장:  나는 사람들은 이상 게 좋아
정답 문장: 그 사람은 자폐성향이 있어
-----------------------------------
입력 문장: be careful
번역한 문장:  제발
정답 문장: 조심해
-----------------------------------
입력 문장: tom lost
번역한 문장:  톰이 자기 말을 껐어
정답 문장: 톰이 졌어
-----------------------------------
입력 문장: i think tom is lonely
번역한 문장:  톰이 자기 말을 껐어
정답 문장: 톰이 외로워하는 것 같아
-----------------------------------
입력 문장: the material is light enough to float in water
번역한 문장:  그만 와
정답 문장: 이 물질은 물에 뜰 정도로 가벼워
-----------------------------------
입력 문장: i had a weird dream last night
번역한 문장:  난 내 최종 목표가 수 않아
정답 문장: 나 어젯밤 이상한 꿈을 꿨어
```

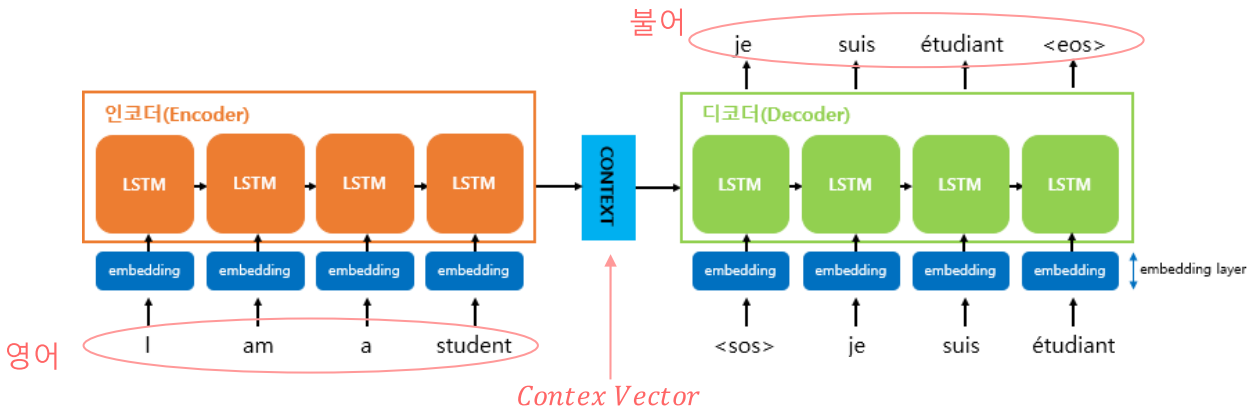
Attention in Word-level

... english to korean ....

Attention Translator – English-Korean

seq2seq 모델의 단점

- 인코더에서 입력 sequence 전체에 대한 정보를 context vector 한개에 압축하여 디코더로 전달
- 디코더는 context vector에 의존하여 출력 Sequence 생성
- 하나의 context vector에 모든 정보를 담아야 하므로 정보 손실 발생 (정확성 저하)
- RNN의 근본적인 약점인 Vanishing Gradient (기울기 소실) 문제 상존



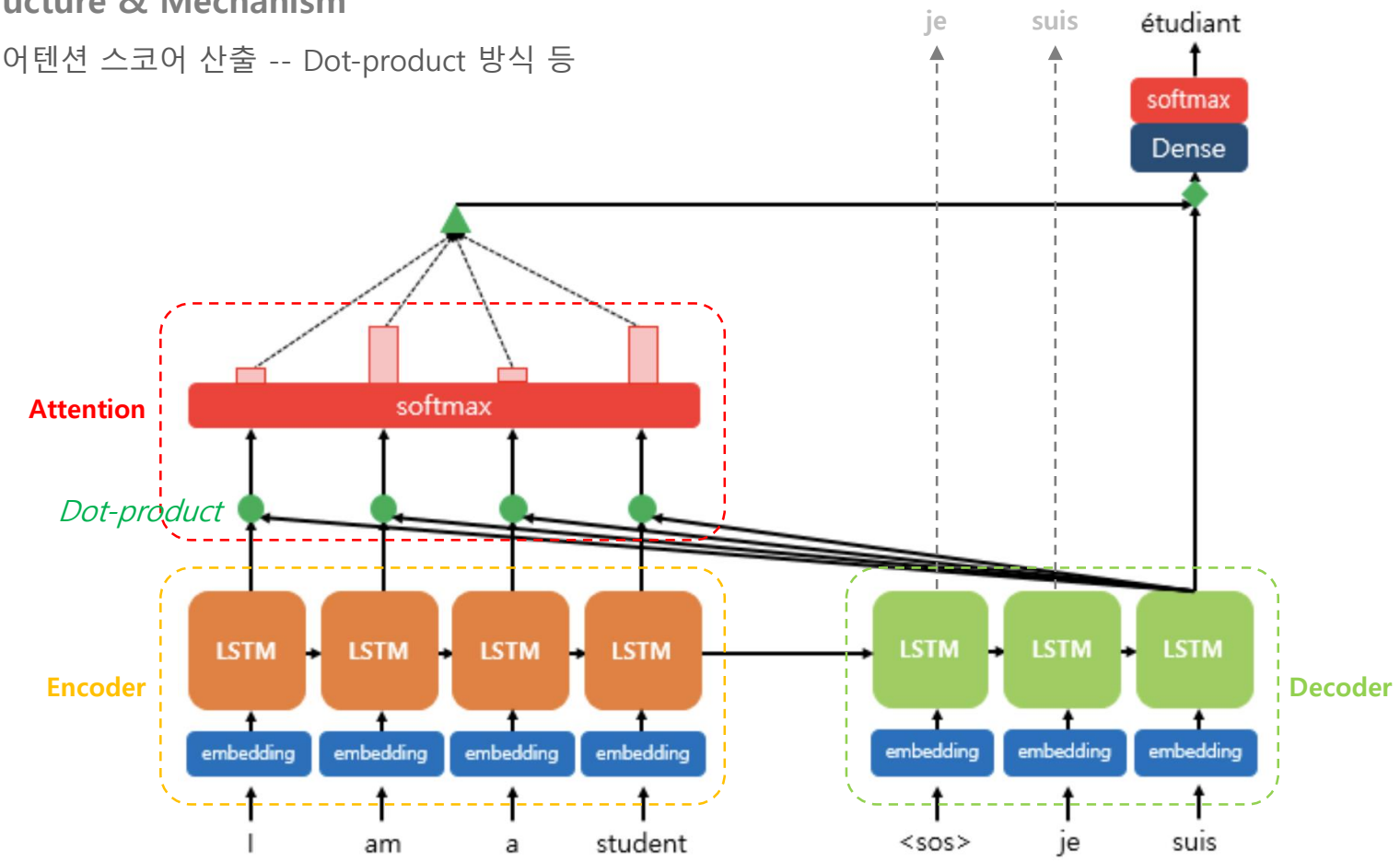
Attention 개념

- 디코더에서 출력 단어를 예측하는 매 시점마다 인코더에서의 전체 입력 문장을 다시 한 번 참고
- 각 시점에서 예측해야할 출력 단어와 연관있는 입력 단어를 집중(어텐션)하여 참고함
  (모든 입력단어를 동일한 비율로 참고하는 것이 아님)

Attention Translator – English-Korean

Structure & Mechanism

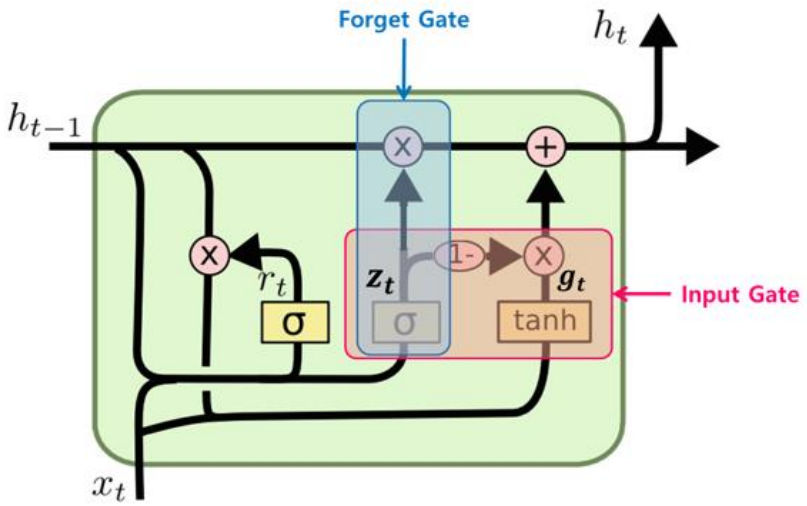
- 어텐션 스코어 산출 -- Dot-product 방식 등



Attention Translator – English-Korean

GRU Cell

- LSTM 셀의 간소화된 버전 -- 2014년 조경현박사(현재 뉴욕대교수, Facebook AI Research 멤버) 제안



- LSTM Cell에서의 두 상태 벡터 c_t와 h_t가 하나의 벡터 h_t로 합쳐짐
- 하나의 gate controller인 z_t가 forget과 input 게이트(gate)를 모두 제어
- z_t가 1을 출력하면 forget 게이트가 열리고 input 게이트가 닫히며, z_t가 0일 경우 반대로 forget 게이트가 닫히고 input 게이트가 열림. 즉, 이전(t-1)의 기억이 저장 될때 마다 타임 스텝 t의 입력은 삭제
- GRU 셀은 output 게이트가 없어 전체 상태 벡터 h_t가 타임 스텝마다 출력되며, 이전 상태 h_{t-1}의 어느 부분이 출력될지를 제어하는 새로운 gate controller인 r_t가 존재함

```
def gru(units):
    return tf.keras.layers.GRU(units, return_sequences=True, return_state=True,
        recurrent_activation='sigmoid',  recurrent_initializer='glorot_uniform')
```

15 Attention Translator – English-Korean

Encoder

- ?

```
class Encoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, enc_units, batch_sz):
        super(Encoder, self).__init__()
        self.batch_sz = batch_sz
        self.enc_units = enc_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.enc_units)

    def call(self, x, hidden):
        x = self.embedding(x)
        output, state = self.gru(x, initial_state = hidden)
        return output, state

    def initialize_hidden_state(self):
        return tf.zeros((self.batch_sz, self.enc_units))
```

15 Attention Translator – English-Korean

Decoder-1/2

- ?

```
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        super(Decoder, self).__init__()
        self.batch_sz = batch_sz
        self.dec_units = dec_units
        self.embedding = tf.keras.layers.Embedding(vocab_size, embedding_dim)
        self.gru = gru(self.dec_units)
        self.fc = tf.keras.layers.Dense(vocab_size)
                # used for attention
        self.W1 = tf.keras.layers.Dense(self.dec_units)
        self.W2 = tf.keras.layers.Dense(self.dec_units)
        self.V = tf.keras.layers.Dense(1)
```

Attention Translator – English-Korean

Decoder-2/2

- ?

```
class Decoder(tf.keras.Model):
    def __init__(self, vocab_size, embedding_dim, dec_units, batch_sz):
        . . .
    def call(self, x, hidden, enc_output):
        hidden_with_time_axis = tf.expand_dims(hidden, 1)
        score = tf.nn.tanh(self.W1(enc_output) + self.W2(hidden_with_time_axis))
        attention_weights = tf.nn.softmax(self.V(score), axis=1)

        context_vector = attention_weights * enc_output
        context_vector = tf.reduce_sum(context_vector, axis=1)

        x = self.embedding(x)
        x = tf.concat([tf.expand_dims(context_vector, 1), x], axis=-1)

        output, state = self.gru(x)
        output = tf.reshape(output, (-1, output.shape[2]))

        x = self.fc(output)

        return x, state, attention_weights
```

17 Attention Translator – English-Korean

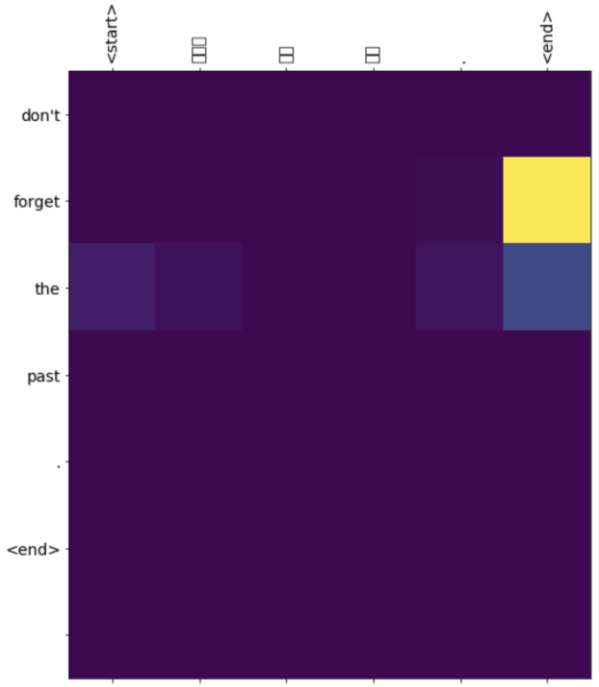
Translate Function

```
def translate(sentence, encoder, decoder,
              inp_lang, targ_lang, max_length_inp, max_length_targ):

    result, sentence, attention_plot = evaluate(
                                            sentence, encoder, decoder,
                            inp_lang, targ_lang, max_length_inp, max_length_targ)

    print('Input: {}'.format(sentence))
    print('Predicted translation: {}'.format(result))

    attention_plot = attention_plot[:len(result.split(' ')),
                                    :len(sentence.split(' '))]

    plot_attention(attention_plot, sentence.split(' '), result.split(' '))
```

Attention Translator – English-Korean

A few good results

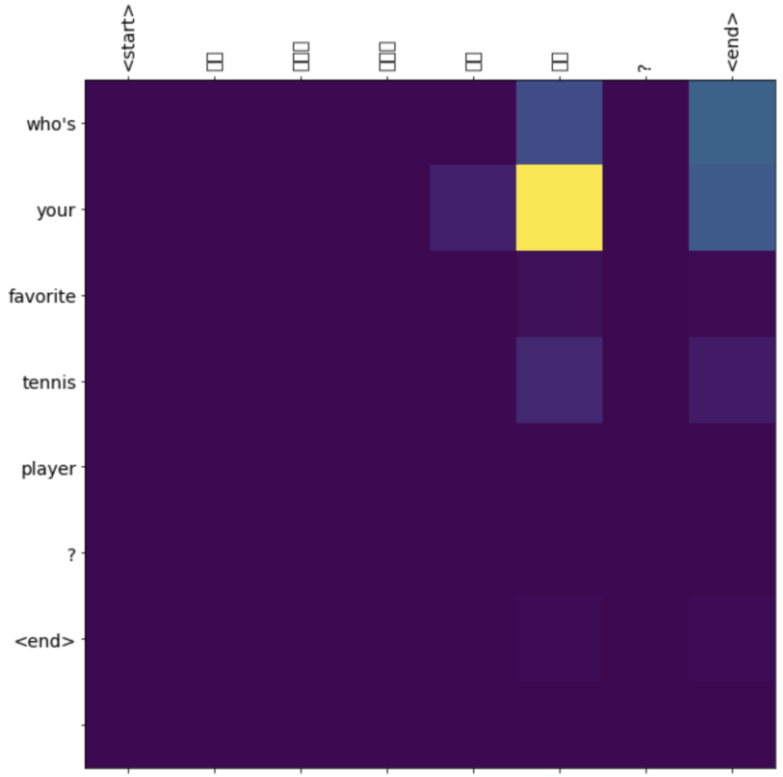
```
Input: <start> 과거를 잊지 말아 . <end>
Predicted translation: don't forget the past . <end>
```



Attention Translator – English-Korean

A few good results

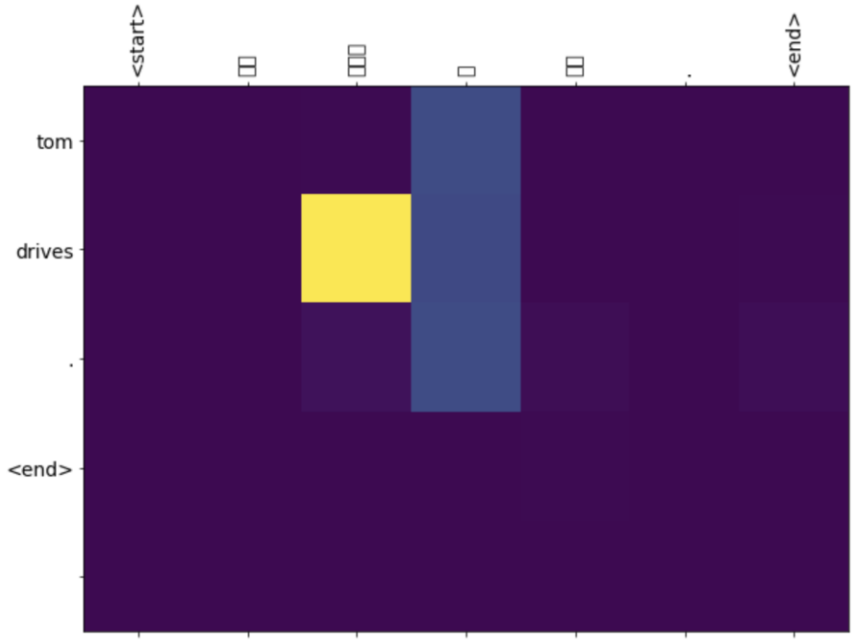
```
Input: <start> 어떤 테니스 선수가 가장 좋아 ? <end>
Predicted translation: who's your favorite tennis player ? <end>
```



17 Attention Translator – English-Korean

A few good results

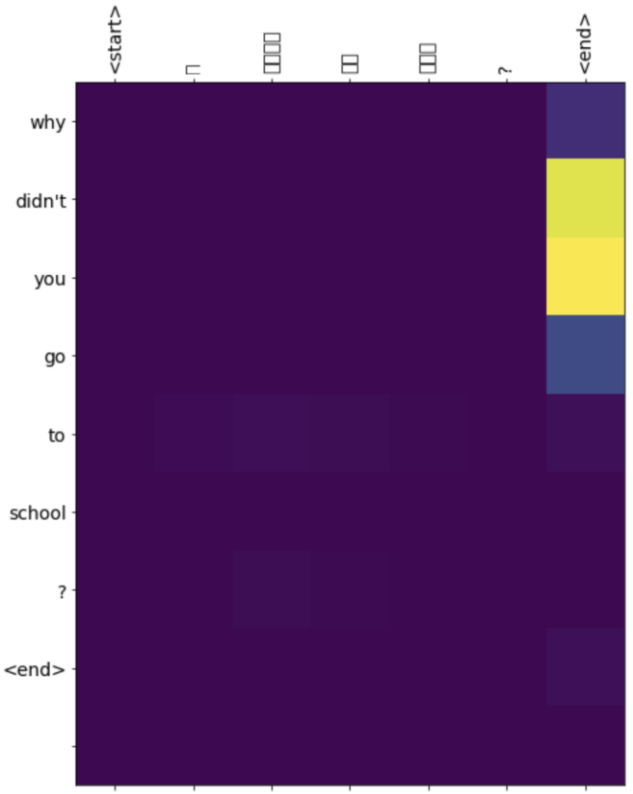
```
Input: <start> 톰은 운전할 수 있어 . <end>
Predicted translation: tom drives . <end>
```



Attention Translator – English-Korean

Make sense but wrong

```
Input: <start> 왜 경찰서에 가지 않았어 ? <end>
Predicted translation: why didn't you go to school ? <end>
```



Attention Translator – English-Korean

Make sense but wrong

```
Input: <start> 거기서 얼마나 기다리고 있을 거야 ? <end>
Predicted translation: how long can you call ? <end>
```