Neural Machine Translation

Bok, Jong Soon
javaexpert@nate.com
https://github.com/swacademy

Contents

- Sequence-to-Sequence
- Attention Mechanism

I. Sequence-to-sequence

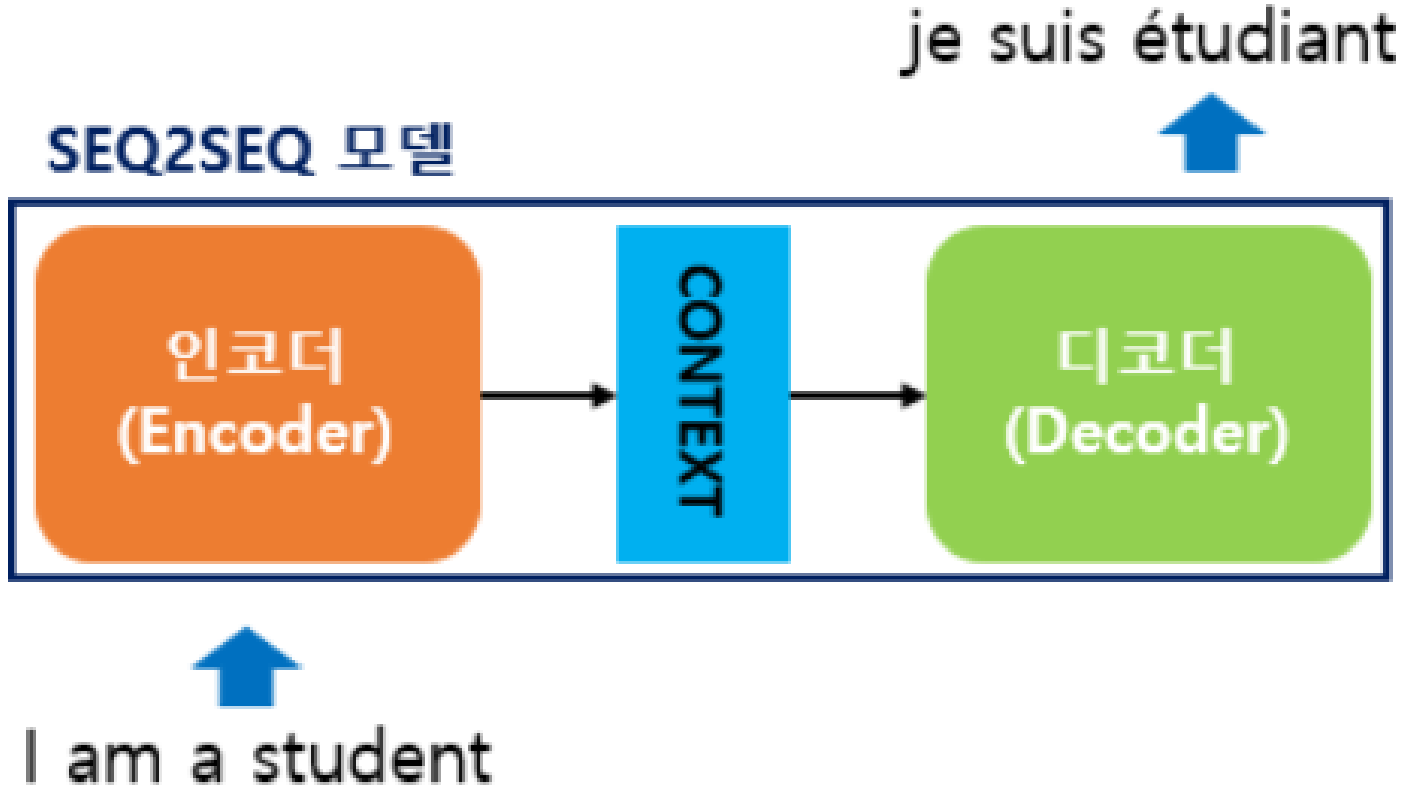
1. seq2seq

시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq)

- 입력된 Sequence로부터 다른 Domain의 Sequence를 출력하는 다양한 분야에서 사용되는 Model이다.
- 예)
  - Chatbot
    - 입력 Sequence와 출력 Sequence를 각각 질문과 대답으로 구성
  - Machine Translation
    - 입력 Sequence와 출력 Sequence를 각각 입력 문장과 번역 문장으로 만듦
  - 내용 요약(Text Summarization)
  - STT(Speech to Text)

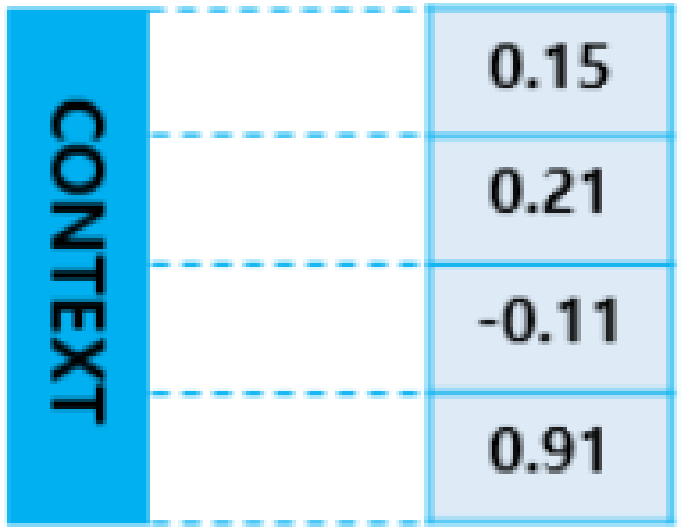
시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

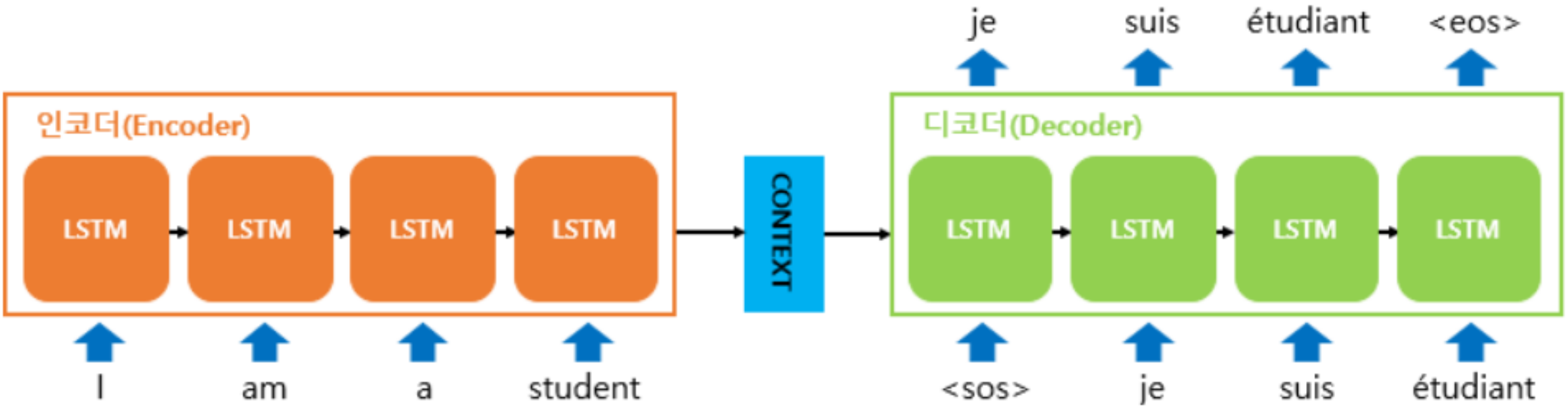


시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

- seq2seq의 2개의 Architecture
- Encoder
  - Context Vector 생성
- Decoder
  - Context vector를 번역된 단어를 한 개씩 순차적으로 출력
- Context Vector
  - Encoder는 입력 문장의 모든 단어들을 순차적으로 입력 받은 뒤에 마지막에 이 모든 단어 정보들을 압축해서 하나의 Vector로 만듦
- 입력 문장의 정보가 하나의 Context Vector로 모두 압축되면 Encoder는 Context Vector를 Decoder로 전송



시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)



시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

- Encoder Architecture와 Decoder Architecture의 내부는 두 개의 RNN Architecture이다.
- Encoder는 입력 문장을 받는 RNN Cell이고, Decoder는 출력 문장을 출력하는 RNN Cell이다.
- 성능 문제로 인해 실제로는 **Vanilla RNN이 아니라 LSTM Cell 또는 GRU Cell들로 구성**

시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

- **Encoder**
  - 입력 문장은 단어 Token化를 통해서 단어 단위로 쪼개지고 단어 Token 각각은 RNN Cell의 각 시점의 입력이 된다.
  - Encoder RNN Cell은 모든 단어를 입력 받은 뒤에 **Encoder RNN Cell의 마지막 시점의 은닉 상태**를 Decoder RNN Cell로 넘겨준다.
  - 이것을 **Context Vector**라고 한다.
  - Context Vector는 **Decoder RNN Cell의 첫번째 은닉 상태**로 사용된다.

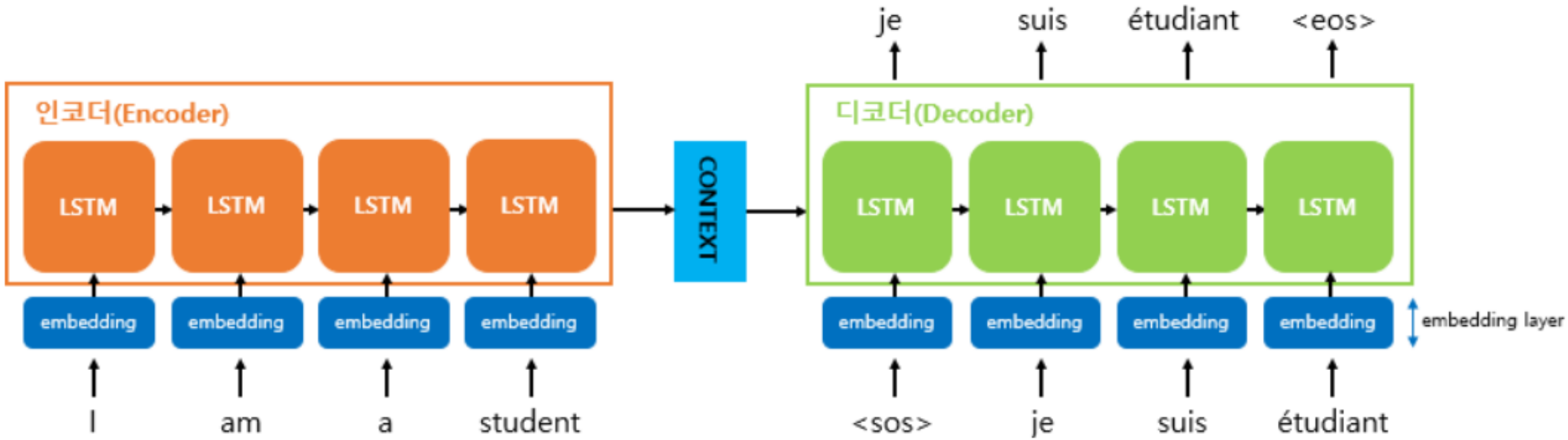
시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

- **Decoder**
  - 기본적으로 RNNLM(RNN Language Model)이다.
  - 초기 입력으로 문장의 시작을 의미하는 Symbol <sos>가 들어간다.
  - <sos>가 입력되면, 다음에 등장할 확률이 높은 단어를 예측한다.
  - 첫 번째 시점(time step)의 Decoder RNN Cell은 다음에 등장할 단어로 je를 예측했다.
  - 첫 번째 시점의 Decoder RNN Cell은 예측된 단어 je를 다음 시점의 RNN Cell의 입력으로 입력한다.
  - 두 번째 시점의 Decoder RNN Cell은 입력된 단어 je로부터 다시 다음에 올 단어인 suis를 예측하고, 또 다시 이것을 다음 시점의 RNN Cell의 입력으로 보낸다.
  - 이런 식으로 기본적으로 다음에 올 단어를 예측하고, 그 예측한 단어를 다음 시점의 RNN Cell의 입력으로 넣는 행위를 반복한다.
  - 이 행위는 문장의 끝을 의미하는 Symbol인 <eos>가 다음 단어로 예측될 때까지 반복된다.

시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

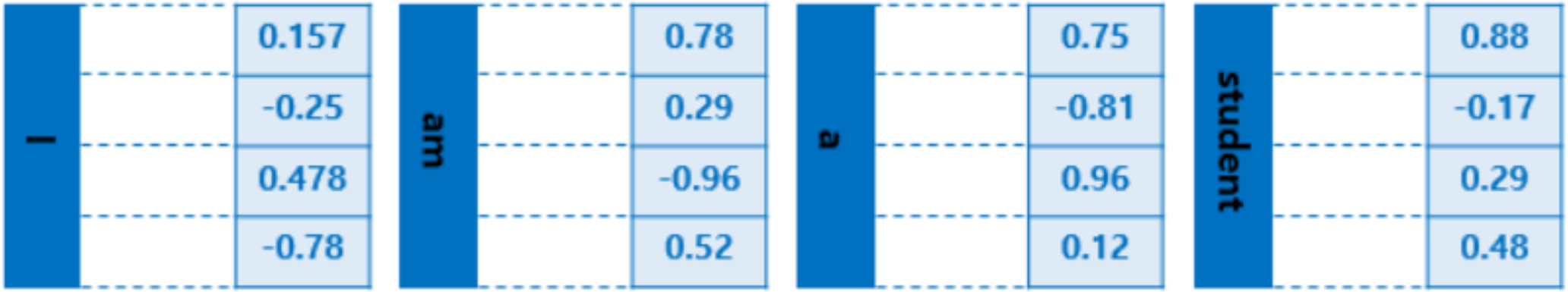
- seq2seq는 훈련 과정과 Test 과정(또는 실제 번역기를 사람이 쓸 때)의 작동 방식이 조금 다르다.
- 훈련 과정에서는 Decoder에게 Encoder가 보낸 Context Vector와 실제 정답 상황인 **<sos> je suis étudiant**를 입력 받았을 때, **je suis étudiant <eos>**가 나와야 된다고 정답을 알려주면서 훈련한다.
- 반면 Test 과정에서는 앞서 설명한 과정과 같이 Decoder는 오직 **Context Vector**와 **<go>**만을 입력으로 받은 후에 다음에 올 단어를 예측하고, 그 단어를 다음 시점의 RNN Cell의 입력으로 넣는 행위를 반복한다.

시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)



시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

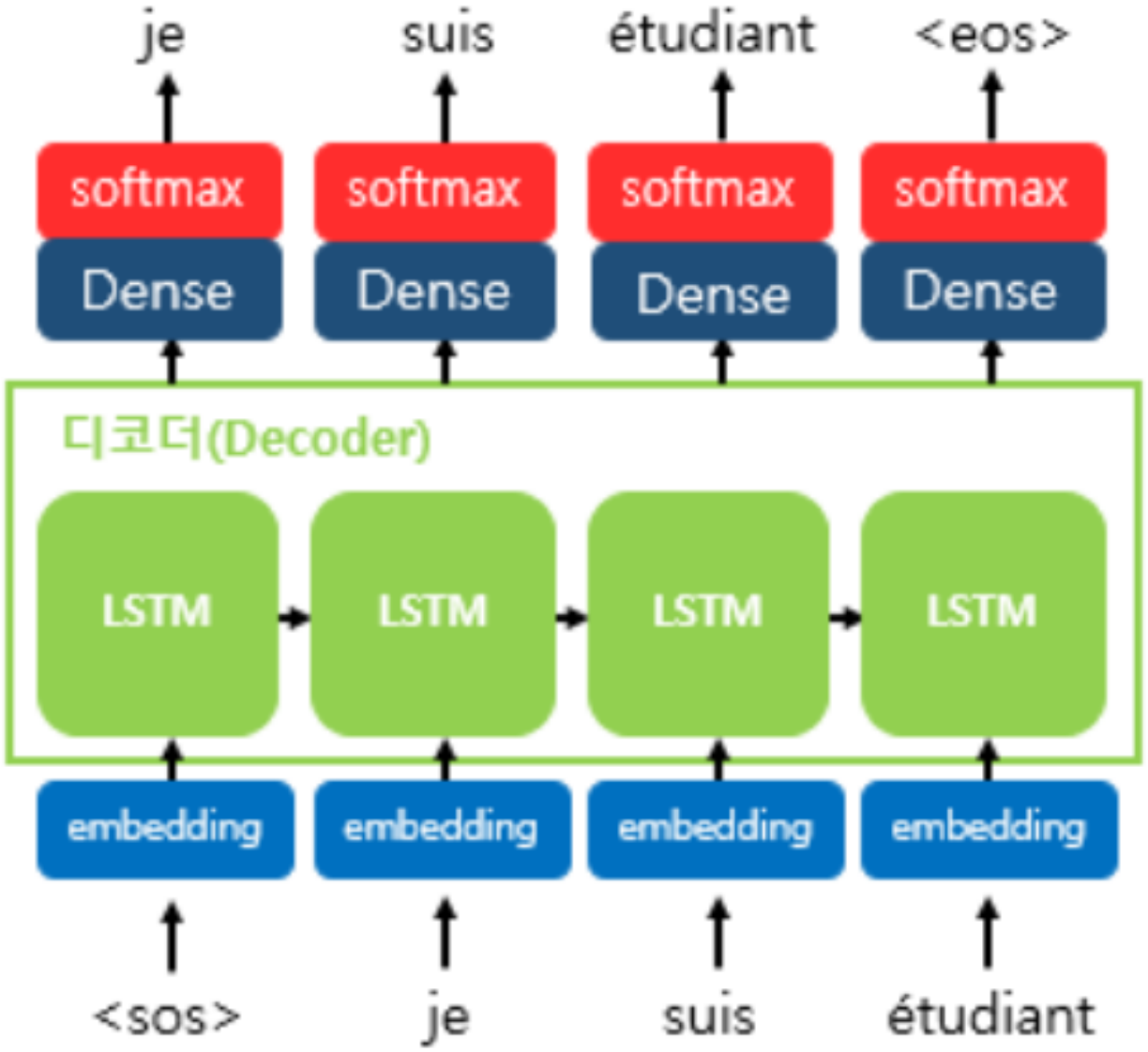
- seq2seq에서 사용되는 모든 단어들은 Word Embedding을 통해 표현된 Embedding Vector이다.
- 앞의 그림은 모든 단어에 대해서 Embedding 과정을 거치게 하는 단계인 Embedding Layer의 모습이다.



시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

- Decoder는 Encoder의 마지막 RNN Cell의 은닉 상태인 Context Vector를 첫 번째 은닉 상태의 값으로 사용한다.
- Decoder의 첫 번째 RNN Cell은 이 첫 번째 은닉 상태의 값과, 현재 t에서의 입력값인 **<sos>**로부터, 다음에 등장할 단어를 예측한다.
- 그리고 이 예측된 단어는 다음 시점인 t+1 RNN에서의 입력값이 되고, 이 t+1에서의 RNN 또한 이 입력값과 t에서의 은닉 상태로부터 t+1에서의 출력 Vector 즉, 또 다시 다음에 등장할 단어를 예측하게 될 것이다.

시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)



시퀀스-투-시퀀스(Sequence-to-Sequence, seq2seq) (Cont.)

- 출력 단어로 나올 수 있는 단어들은 다양한 단어들이 있다.
- seq2seq Model은 선택될 수 있는 모든 단어들로부터 하나의 단어를 골라서 예측해야 한다.
- 이를 예측하기 위해서 쓸 수 있는 함수로 Softmax 함수가 있다.
- Decoder에서 각 시점(time step)의 RNN Cell에서 출력 Vector가 나오면, 해당 Vector는 Softmax 함수를 통해 출력 Sequence의 각 단어별 확률값을 반환하고, Decoder는 출력 단어를 결정한다.

2. Character-Level Neural Machine Translation

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기

- Keras 프랑수아 숄레의 Blog의 유명 게시물인 'sequence-to-sequence 10 분만에 이해하기'
- Link : https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html
- 기계 번역기를 훈련시키기 위해서는 훈련 데이터로 병렬 코퍼스(parallel corpus)가 필요하다.
- 병렬 Corpus
  - 두 개 이상의 언어가 병렬적으로 구성된 Corpus를 의미.
- Download Link : http://www.manythings.org/anki
- 프랑스-영어 병렬 Corpus(fra-eng.zip)
  - fra.txt

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리

- 기본적으로 입력 Sequence와 출력 Sequence의 길이는 다를 수 있다.
- 실제 번역기를 생각해보면 Google 번역기에 ' 나는 학생이다. ' 라는 Token의 개수가 2인 문장을 넣었을 때 →
  - 'I am a student.'라는 Token의 개수가 4인 문장이 나오는 것과 같은 이치이다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

Watch me. Regardez-moi !

- 사용할 fra.txt Data는 왼쪽의 영어 문장과 오른쪽의 프랑스어 문장 사이에 탭으로 구분되는 구조가 하나의 Sample이다.
- 이와 같은 형식의 약 17만개의 병렬 문장 Sample을 포함하고 있다.

```
1  import pandas as pd
2  lines= pd.read_csv('./fra.txt', names=['src', 'tar'], sep='₩t')
3  len(lines)
```

```
170190
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 해당 Data는 약 17만여개의 병렬 문장 Sample로 구성되어 있지만 여기서는 간단히 60,000개의 Sample만 가지고 기계 번역기를 구축해보도록 한다.
- 전체 Data 중 60,000개의 Sample만 저장하고 현재 Data가 어떤 구성이 되었는지 확인해보자.

```
lines = lines[0:60000] # 6만개만 저장
lines.sample(10)
```

	src	tar
39776	Here's a photo of her.	Voilà une photo d'elle.
12736	What did you do?	Qu'avez-vous fait ?
42154	No one recognized you.	Personne ne vous a reconnus.
56904	Which team won the game?	Quelle équipe l'a emporté ?
28285	I can come at three.	Ok pour 3 heures.
25773	Tom is John's twin.	Tom est le jumeau de John.
5962	It's a rental.	C'est une location.
40853	I talked to everybody.	Je parlai avec tout le monde.
43792	What are they made of?	De quoi sont-elles faites ?
7656	I do apologize.	Je présente vraiment mes excuses.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 번역 문장에 해당되는 French Data는 앞서 배웠듯이 시작을 의미하는 Symbol **<sos>**과 종료를 의미하는 Symbol **<eos>**을 넣어주어야 한다.
- 여기서는 **<sos>**와 **<eos>** 대신 **'\t'**를 시작 Symbol, **'\n'**을 종료 Symbol로 간주하여 추가하고 다시 Data를 출력해보자.

```
1 lines.tar = lines.tar.apply(lambda x : '₩t '+ x + ' ₩n')
2 lines.sample(10)
```

	src	tar
58516	Have you eaten lunch yet?	\t Avez-vous déjà dîné ? \n
49990	We know you're in pain.	\t Nous savons que vous avez mal. \n
5217	I cut classes.	\t J'ai séché les cours. \n
34844	I want to go fishing.	\t Je veux aller pêcher. \n
735	I'm naked.	\t Je me trouve nu. \n
15245	It's bad for you.	\t C'est mauvais pour vous. \n
33169	Everyone was stunned.	\t Tout le monde fut paralysé. \n
11258	I'm not wealthy.	\t Je ne suis pas riche. \n
53020	I have a bath every day.	\t Je prends un bain tous les jours. \n
2130	I could try.	\t Je pourrais essayer. \n

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 글자 집합을 생성한다.
- 단어 집합이 아니라 글자 집합이라고 하는 이유는 Token 단위가 단어가 아니라 글자이기 때문이다.

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

tar_vocab=set()
for line in lines.tar:
    for char in line:
        tar_vocab.add(char)
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 글자 집합의 크기를 보자.

```
1 src_vocab_size = len(src_vocab)+1
2 tar_vocab_size = len(tar_vocab)+1
3 print(src_vocab_size)
4 print(tar_vocab_size)
```

```
79
106
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 영어와 프랑스어는 각각 약 80개와 100개의 글자가 존재한다.
- 이 중에서 Index를 임의로 부여하여 일부만 출력해본다.
- 현 상태에서 Index를 사용하려고 하면 Error가 발생한다.
- 하지만 정렬하여 순서를 정해준 뒤에 Index를 사용하여 출력하면 된다.

```
1 src_vocab = sorted(list(src_vocab))
2 tar_vocab = sorted(list(tar_vocab))
3 print(src_vocab[45:75])
4 print(tar_vocab[45:75])
```

```
['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']
['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']
```

- 글자 집합에 글자 단위로 저장된 것을 확인할 수 있다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 각 글자에 Index를 부여한다.

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

```
{' ': 1, '!': 2, '"': 3, '$': 4, '%': 5, '&': 6, "'": 7, ',': 8, '-': 9, '.': 10, '/': 11, '0': 12, '1': 13, '2': 14, '3': 15, '4': 1
6, '5': 17, '6': 18, '7': 19, '8': 20, '9': 21, ':': 22, '?': 23, 'A': 24, 'B': 25, 'C': 26, 'D': 27, 'E': 28, 'F': 29, 'G': 30, 'H':
31, 'I': 32, 'J': 33, 'K': 34, 'L': 35, 'M': 36, 'N': 37, 'O': 38, 'P': 39, 'Q': 40, 'R': 41, 'S': 42, 'T': 43, 'U': 44, 'V': 45,
'W': 46, 'X': 47, 'Y': 48, 'Z': 49, 'a': 50, 'b': 51, 'c': 52, 'd': 53, 'e': 54, 'f': 55, 'g': 56, 'h': 57, 'i': 58, 'j': 59, 'k': 6
0, 'l': 61, 'm': 62, 'n': 63, 'o': 64, 'p': 65, 'q': 66, 'r': 67, 's': 68, 't': 69, 'u': 70, 'v': 71, 'w': 72, 'x': 73, 'y': 74, 'z':
75, 'é': 76, '' ': 77, '€': 78}
{'₩t': 1, '₩n': 2, ' ': 3, '!': 4, '"': 5, '$': 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': 4
5, '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, 'Ć': 100, '₩u2009': 101, '₩u200b': 10
2, '‘': 103, '’': 104, '₩u202f': 105}
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- Index가 부여된 글자 집합으로부터 갖고 있는 훈련 Data에 Integer Encoding을 수행한다.
- Encoder의 입력이 될 영어 문장 Sample에 대해서 Integer Encoding을 수행해보고, 5개의 Sample을 출력해본다.

```
encoder_input = []
for line in lines.src: #입력 데이터에서 1줄씩 문장을 읽음
    temp_X = []
    for w in line: #각 줄에서 1개씩 글자를 읽음
      temp_X.append(src_to_index[w]) # 글자를 해당되는 정수로 변환
    encoder_input.append(temp_X)
print(encoder_input[:5])
```

```
[[30, 64, 10], [31, 58, 10], [31, 58, 10], [41, 70, 63, 2], [41, 70, 63, 2]]
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- Integer Encoding이 끝나면 Decoder의 입력이 될 French Data에 대해서 Integer Encoding을 수행한다.

```
decoder_input = []
for line in lines.tar:
    temp_X = []
    for w in line:
      temp_X.append(tar_to_index[w])
    decoder_input.append(temp_X)
print(decoder_input[:5])
```

```
[[1, 3, 48, 53, 3, 4, 3, 2], [1, 3, 45, 53, 64, 73, 72, 3, 4, 3, 2], [1, 3, 45, 53, 64, 73, 72, 14, 3, 2], [1, 3, 29, 67, 73, 70, 71, 105, 4, 3, 2], [1, 3, 29, 67, 73, 70, 57, 78, 105, 4, 3, 2]]
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 아직 정수 Encoding을 수행해야 할 Data가 하나 더 남아 있다.
- Decoder의 예측값과 비교하기 위한 실제값이 필요하다.
- 그런데 이 실제값에는 시작 심볼에 해당되는 **<sos>**가 있을 필요가 없다.
- 그래서 이번에는 정수 Encoding 과정에서 **<sos>**를 제거한다.
- 즉, 모든 프랑스어 문장의 맨 앞에 붙어있는 **'\t'**를 제거한다.

```
decoder_target = []
for line in lines.tar:
    t=0
    temp_X = []
    for w in line:
      if t>0:
        temp_X.append(tar_to_index[w])
      t=t+1
    decoder_target.append(temp_X)
print(decoder_target[:5])
```

```
[[3, 48, 53, 3, 4, 3, 2], [3, 45, 53, 64, 73, 72, 3, 4, 3, 2], [3, 45, 53, 64, 73, 72, 14, 3, 2], [3, 29, 67, 73, 70, 71, 105, 4, 3, 2], [3, 29, 67, 73, 70, 57, 78, 105, 4, 3, 2]]
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 이제 모든 Data에 대해서 Padding 작업을 수행한다.
- Padding을 위해서 영어 문장과 프랑스어 문장 각각에 대해서 가장 길이가 긴 Sample의 길이를 알아보자.

```
max_src_len = max([len(line) for line in lines.src])
max_tar_len = max([len(line) for line in lines.tar])
print(max_src_len)
print(max_tar_len)
```

```
25
76
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- English Data는 English Sample들끼리, French는 French Sample들끼리 길이를 맞추어서 Padding한다.
- English Data는 English Sample의 가장 긴 길이 25에 맞추어 Padding하고, French Data는 76에 맞춰서 Padding한다.

```
from keras.preprocessing.sequence import pad_sequences
encoder_input = pad_sequences(encoder_input, maxlen=max_src_len, padding='post')
decoder_input = pad_sequences(decoder_input, maxlen=max_tar_len, padding='post')
decoder_target = pad_sequences(decoder_target, maxlen=max_tar_len, padding='post')
```

```
Using TensorFlow backend.
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- Data에 대한 이해와 전처리 (Cont.)

- 모든 값에 대해서 One-hot Encoding을 수행한다.
- 글자 단위 번역기므로 Word Embedding은 별도로 사용되지 않으며, 예측값과의 오차 측정에 사용되는 실제값뿐만 아니라 입력값도 One-hot Vector를 사용하도록 한다.

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

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- 교사 강요(Teacher forcing)

- 훈련 과정에서 이전 시점의 Decoder Cell의 출력을 현재 시점의 Decoder Cell의 입력으로 넣어주지 않고, 이전 시점의 실제값을 현재 시점의 Decoder Cell의 입력값으로 하는 방법을 사용할 것이다.
- 그 이유는 이전 시점의 Decoder Cell의 예측이 틀렸을 때 현재 시점의 Decoder Cell의 예측도 잘못될 가능성이 높고 이는 연쇄 작용으로 Decoder 전체의 예측을 어렵게 만들 것이기 때문이다.
- 그래서 RNN의 모든 시점에 대해서 이전 시점의 예측값 대신 실제값을 입력으로 주는 방법을 교사 강요라고 한다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 훈련시키기

- seq2seq Model을 설계하고 교사 강요를 사용하여 훈련하도록 한다.

```
from keras.layers import Input, LSTM, Embedding, Dense
from 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. 바로 은닉 상태와 셀 상태.
```

- LSTM에서 **state_h**, **state_c**를 리턴받는데, 이는 각각 은닉 상태와 셀 상태에 해당된다.
- 이 두 가지 상태를 **encoder_states**에 저장한다.
- **encoder_states**를 Decoder에 전달함으로 이 두 가지 상태 모두를 Decoder로 전달한다.
- 이것이 **Context Vector**이다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 훈련시키기 (Cont.)

```
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)
# Decoder의 첫 상태를 Encoder의 은닉 상태, Cell 상태로 한다.
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")
```

- Decoder는 Encoder의 마지막 은닉 상태를 초기 은닉 상태로 사용한다.
- 위에서 **initial_state**의 인자값으로 **encoder_states**를 주는 Code가 이에 해당된다.
- 동일하게 Decoder 의 은닉 상태 크기도 256으로 주었다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 훈련시키기 (Cont.)

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

- 입력으로는 Encoder 입력과 Decoder 입력이 들어가고, Decoder의 실제값인 decoder_target도 필요하다.
- Batch size는 64로 하였으며 총 50 Epoch를 학습한다.
- 위에서 설정한 은닉 상태의 크기와 Epoch 수는 실제로는 훈련 Data에 과적합 상태를 불러올 것이다.
- 중간부터 검증 데이터에 대한 오차인 val_loss의 값이 올라가는데, 사실 이번 실습에서는 주어진 Data의 양과 Task의 특성으로 인해 훈련 과정에서 훈련 Data의 정확도와 과적합 방지라는 두 마리 토끼를 동시에 잡기에는 쉽지 않다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 동작시키기

- 전체적인 번역 동작 단계를 정리하면 다음과 같다.
  ① 번역하고자 하는 입력 문장이 Encoder에 들어가서 은닉 상태와 Cell 상태를 얻는다.
  ② 상태와 **<SOS>**에 해당하는 **'\t'**를 Decoder로 보낸다.
  ③ Decoder가 **<EOS>**에 해당하는 **'\n'**이 나올 때까지 다음 문자를 예측하는 행동을 반복한다.

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

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 동작시키기 (Cont.)

```
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)
```

```
index_to_src = dict(
    (i, char) for char, i in src_to_index.items())
index_to_tar = dict(
    (i, char) for char, i in tar_to_index.items())
```

- 단어로부터 Index를 얻는 것이 아니라 Index로부터 단어를 얻을 수 있는 **index_to_src**와 **index_to_tar**를 만든다.

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 동작시키기 (Cont.)

```
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
```

글자 레벨 기계 번역기(Character-Level Neural Machine Translation) 구현하기- seq2seq 기계 번역기 동작시키기 (Cont.)

```
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:  Attends !
-----------------------------------
입력 문장: I lost.
정답 문장:  J'ai perdu.
번역기가 번역한 문장:  Je l'ai vu.
-----------------------------------
입력 문장: Come in.
정답 문장:  Entre !
번역기가 번역한 문장:  Entre !
-----------------------------------
입력 문장: I got it.
정답 문장:  J'ai capté.
번역기가 번역한 문장:  Je l'ai coupablé.
-----------------------------------
입력 문장: What else?
정답 문장:  Quoi d' autre ?
번역기가 번역한 문장:  Qu'est-ce qui s'en soucie ?
```