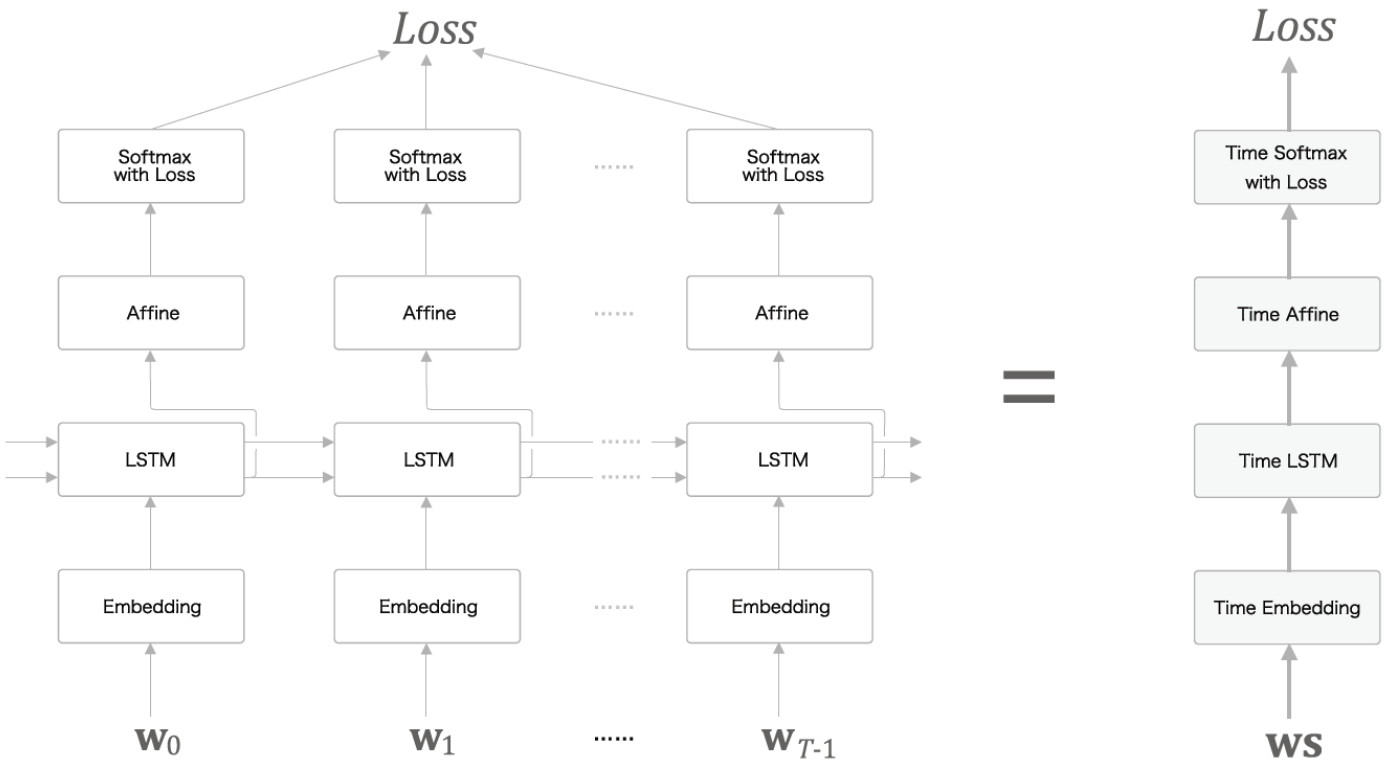
밑바닥부터
시작하는 딥러닝2

7장. RNN을 사용한 문장 생성

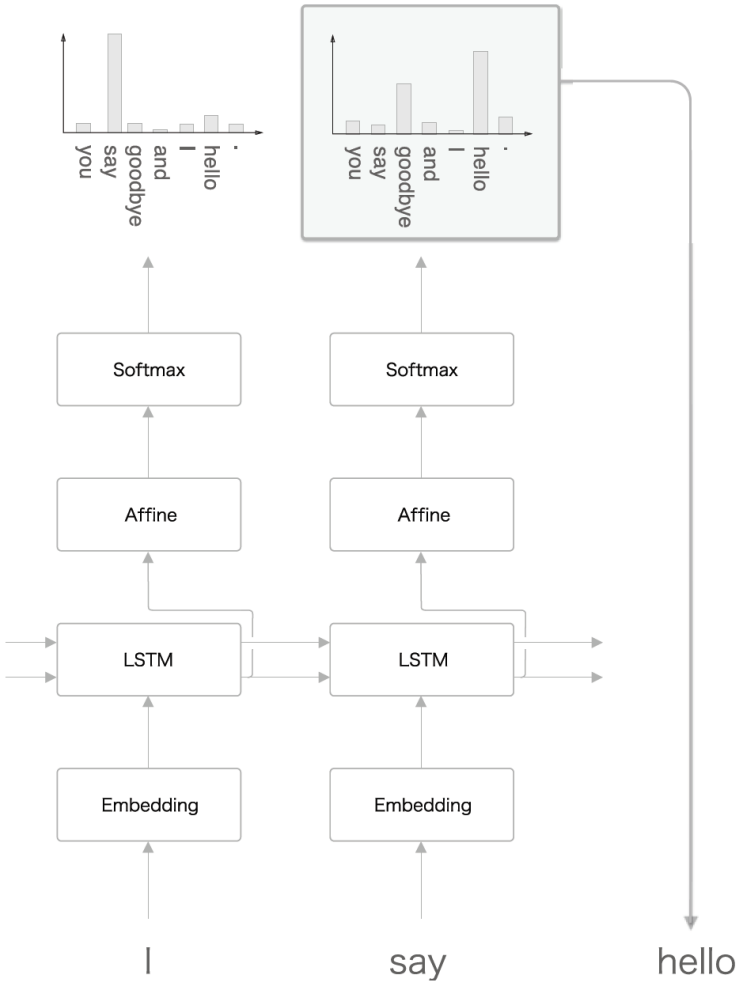
송선영

CONTENTS

01 언어 모델을 사용한 문장 생성



[그림 1]



[그림 2]

- 문장 생성 구현

```
import sys
sys.path.append('..')
import numpy as np
from common.functions import softmax
from ch06.rnnlm import Rnnlm
from ch06.better_rnnlm import BetterRnnlm


class RnnlmGen(Rnnlm):
    def generate(self, start_id, skip_ids=None, sample_size=100):
        word_ids = [start_id]

        x = start_id
        while len(word_ids) < sample_size:
            x = np.array(x).reshape(1, 1)
            score = self.predict(x) # 각 단어의 점수 출력
            p = softmax(score.flatten())  # 출력된 점수를 정규화

            sampled = np.random.choice(len(p), size=1, p=p) # 확률분포에 따라 샘플링
            if (skip_ids is None) or (sampled not in skip_ids):
                x = sampled
                word_ids.append(int(x))

        return word_ids
```

def generate() : 문장 생성하는 메서드

start_id : 최초로 주는 단어의 ID

skip_ids : 단어 ID의 리스트
- 이 리스트에 속하는 단어 ID는 샘플링 되지 않도록 해줌
- ex) <unk>, N

sample_size : 샘플링하는 단어의 수

01 언어 모델을 사용한 문장 생성

```
import sys
sys.path.append('..')
from rnnlm_gen import RnnlmGen
from dataset import ptb


corpus, word_to_id, id_to_word = ptb.load_data('train')
vocab_size = len(word_to_id)
corpus_size = len(corpus)

model = RnnlmGen()
# model.load_params('../ch06/Rnnlm.pkl')

# start 문자와 skip 문자 설정
start_word = 'you'
start_id = word_to_id[start_word]
skip_words = ['N', '<unk>', '$']
skip_ids = [word_to_id[w] for w in skip_words]

# 문장 생성
word_ids = model.generate(start_id, skip_ids)
txt = ' '.join([id_to_word[i] for i in word_ids])
txt = txt.replace(' <eos>', '.₩n')
print(txt)
```

- 학습되지 않은 언어 모델을 사용했을 때

you setback best raised fill steelworkers Montgomery Kohlberg told beam worthy allied ban Swedish aichi mather promptly ramada explicit Leslie bets discovery considering campaigns bottom Petrie warm large-scale frequent temple Grumman bennett …

- 학습된 언어 모델을 사용했을 때

you 'll include one of them a good problems.
moreover so if not gene 's corr experience with the heat of bridges a new deficits model is non-violent what it 's a rule must exploit it.
there 's no tires industry could occur.
beyond my hours where he is n't going home says and Japanese letter.
knight transplants d.c. turmoil with one-third of voters.
the justice department is …

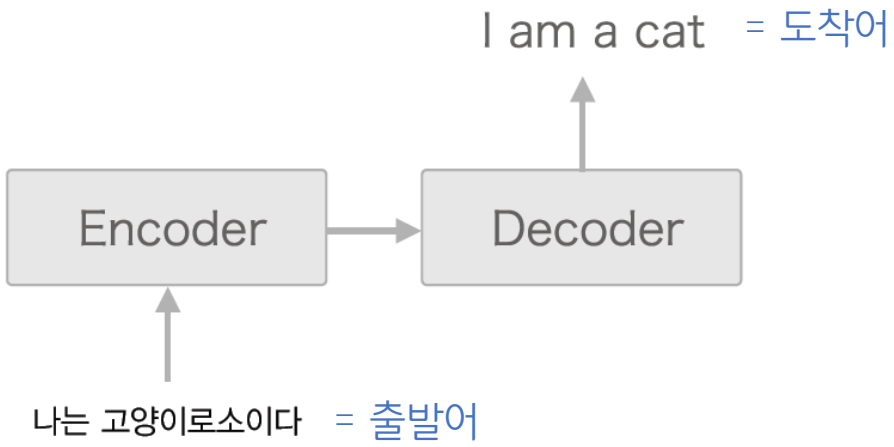
- BetterRnnlm 클래스 사용

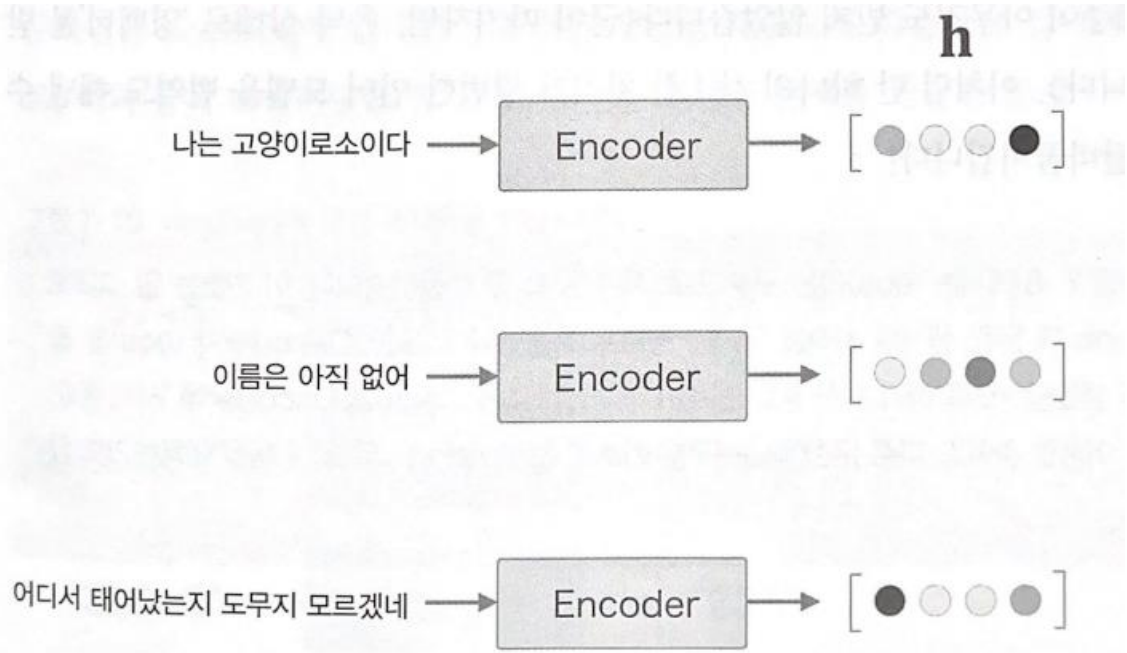
you 've seen two families and the women and two other women of students.
the principles of investors that prompted a bipartisan rule of which had a withdrawn target of black men or legislators interfere with the number of plants can do to carry it together.
the appeal was to deny steady increases in the operation of dna and educational damage in the 1950s.
...

-> 더 좋은 언어 모델을 사용하면 더 좋은 문장을 생성할 수 있다.

더 큰 말뭉치를 사용하면 더 자연스러운 문장을 생성할 수 있다.

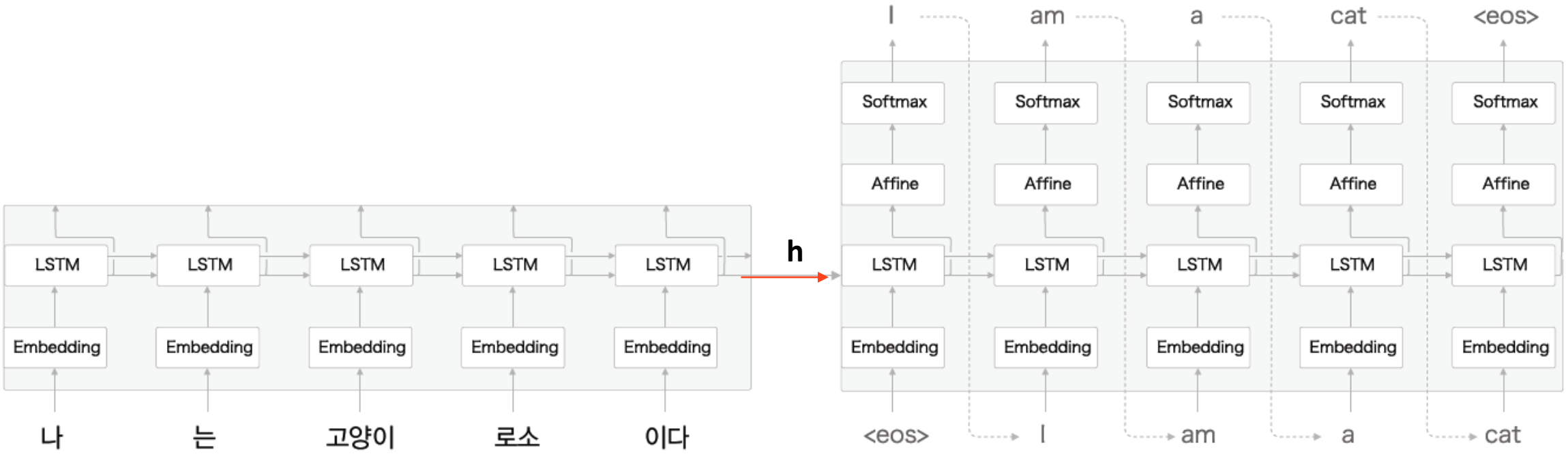
- seq2seq (sequence to sequence)
  - 시계열 데이터를 다른 시계열 데이터로 변환
  - **2개의 RNN** (Encoder, Decoder) 을 이용





-> 문장을 고정 길이 벡터로 인코딩

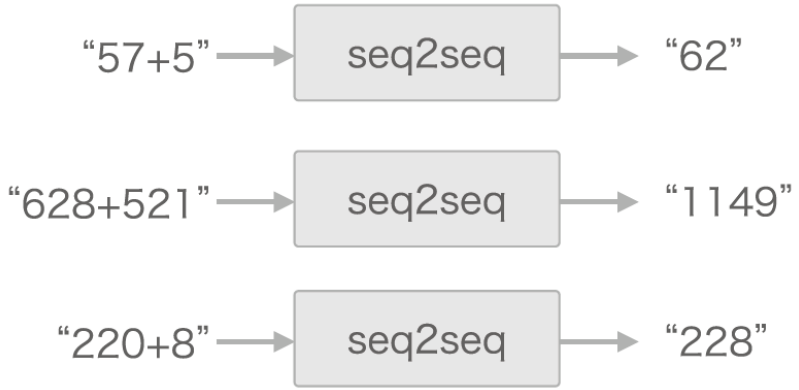
- Seq2seq의 전체 계층 구성



< Encoder >

< Decoder >

- 가변 길이 시계열 데이터 예시



입력

5	7	+	5			
6	2	8	+	5	2	1
2	2	0	+	8		

출력

_	6	2		
_	1	1	4	9
_	2	2	8	

-> 패딩 사용

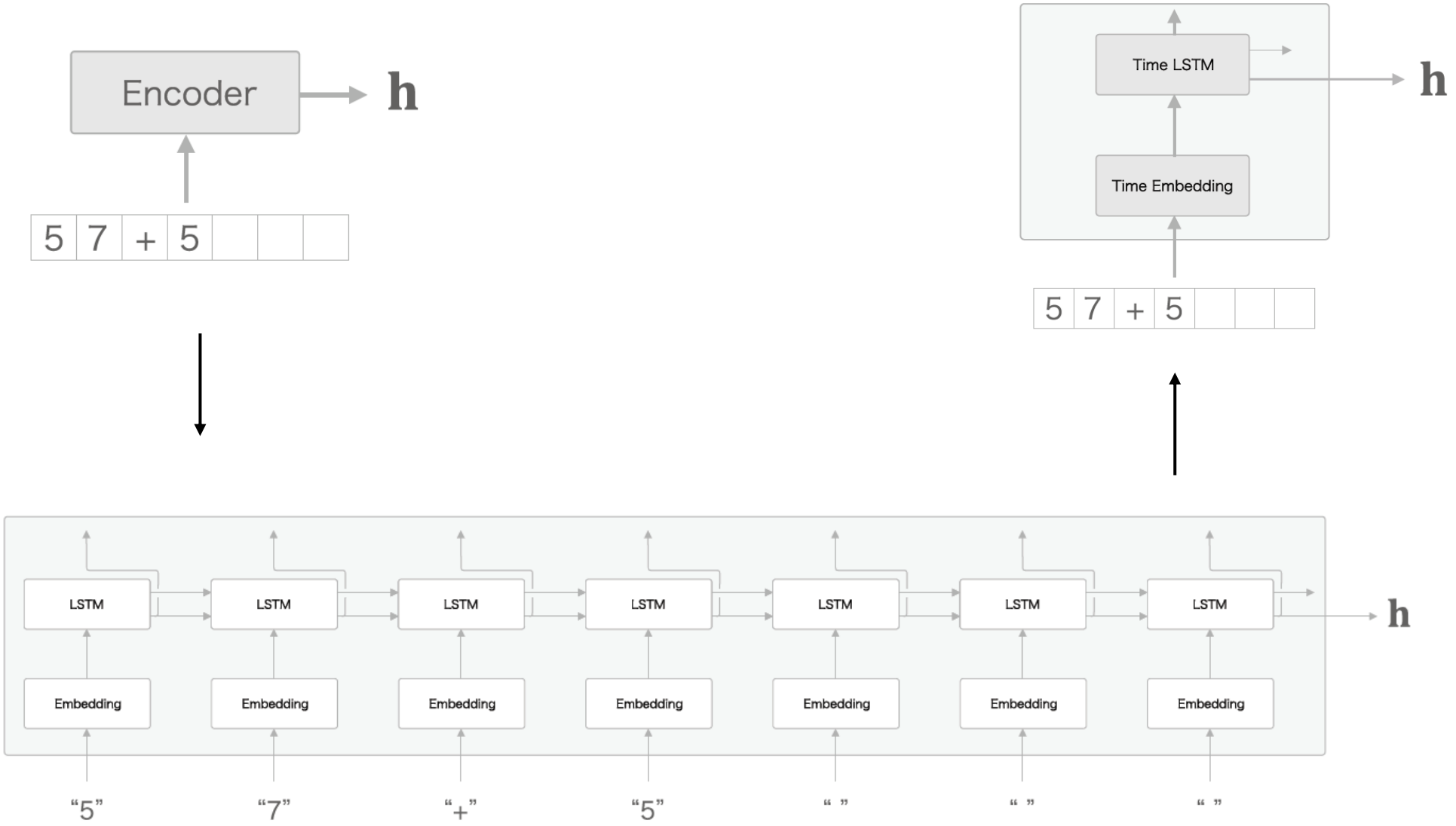
```
import sys
sys.path.append('..')
from dataset import sequence


(x_train, t_train), (x_test, t_test) = ㅐ
    sequence.load_data('addition.txt', seed=1984)
char_to_id, id_to_char = sequence.get_vocab()

print(x_train.shape, t_train.shape)
print(x_test.shape, t_test.shape)
# (45000, 7) (45000, 5)
# (5000, 7) (5000, 5)

print(x_train[0])
print(t_train[0])
# [ 3  0  2  0  0 11  5]
# [ 6  0 11  7  5]

print(''.join([id_to_char[c] for c in x_train[0]]))
print(''.join([id_to_char[c] for c in t_train[0]]))
# 71+118
# _189
```

get_vocab() : 문자와 문자 ID의 대응 관계를 담은 딕셔너리를 반환

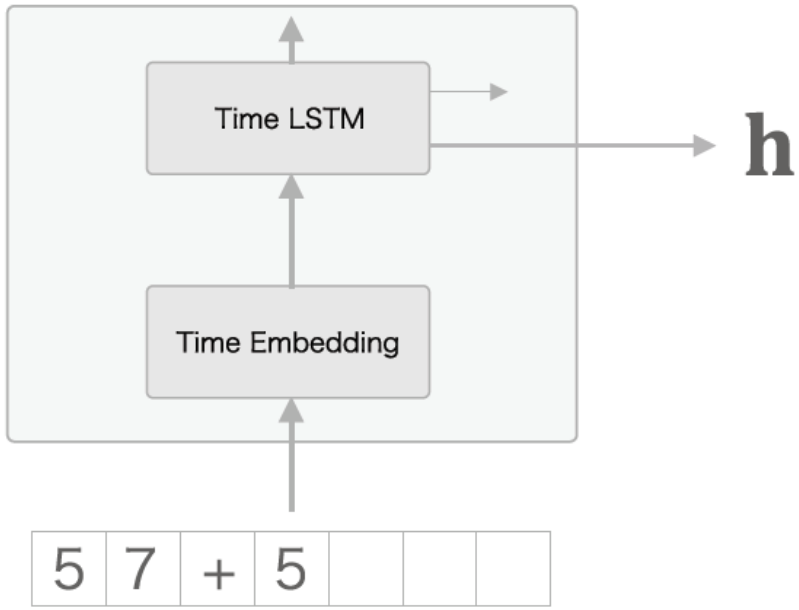
- char_to_id: {'문자' : '문자id'}
- id_to_char: {'문자id' : '문자'}

x_train[0]	id	[3 0 2 0 0 11 5]
		↓
	char	[7 1 + 1 1 8 " "]
t_train[0]	id	[6 0 11 7 5]
		↓
	char	[_ 1 8 9 " "]

- Encoder

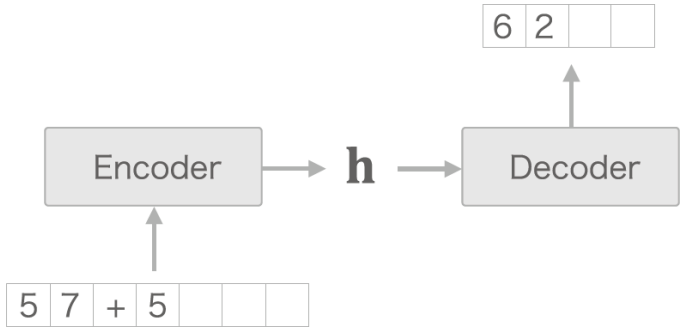

Encoder
h
5 7 + 5
Time LSTM
Time Embedding
h
5 7 + 5
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
LSTM
h
Embedding
Embedding
Embedding
Embedding
Embedding
Embedding
Embedding
“5”
“7”
“+”
“5”
“ ”
“ ”
“ ”

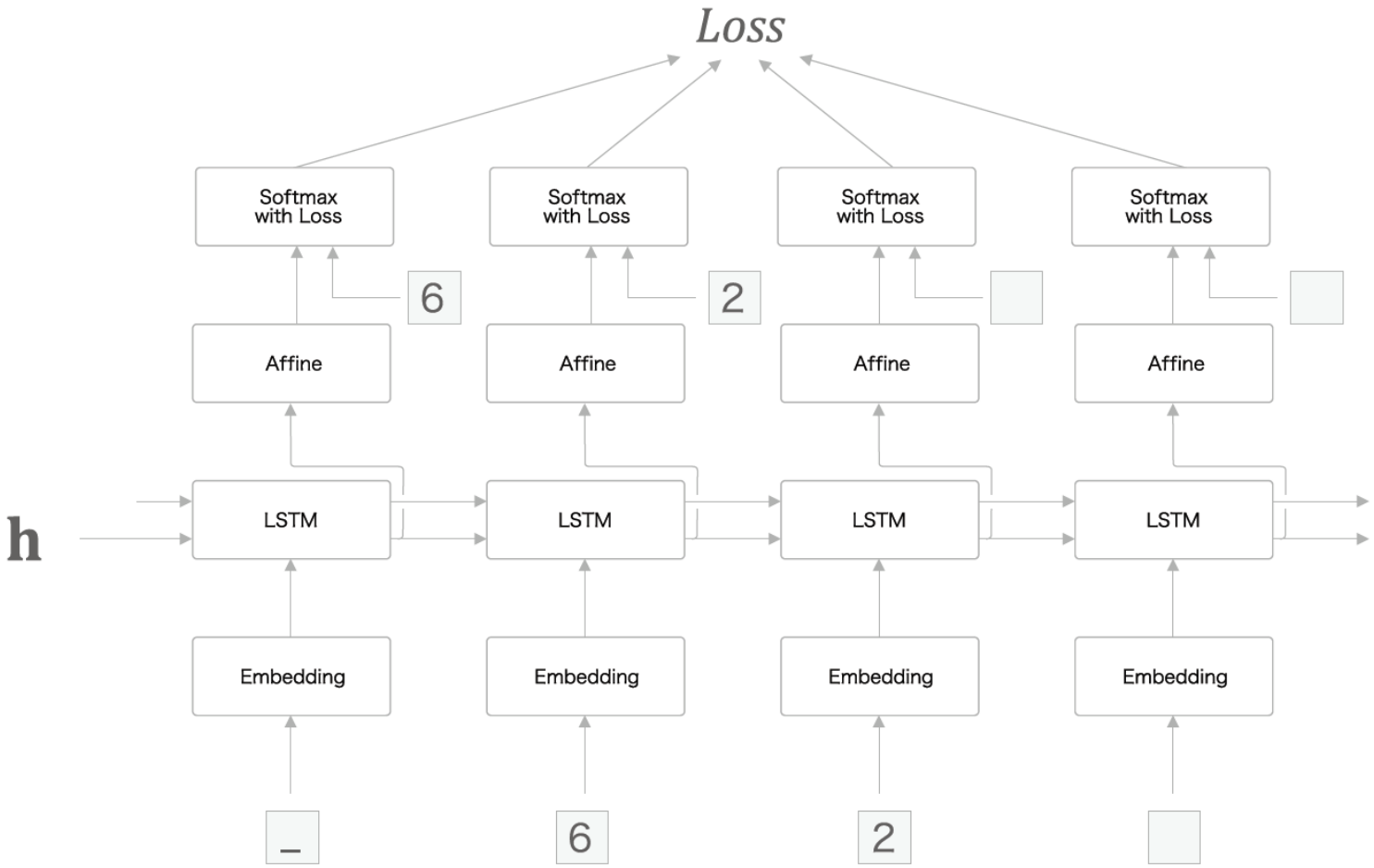
- Encoder 구현




```python
class Encoder:
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        # vocab_size: 어휘 수(문자의 종류)
        #  -> 0~9, '+' , '_', ' ' 로 13가지
        # wordvec_size: 문자 벡터의 차원 수
        # hidden_size: LSTM계층의 은닉 상태 벡터의 차원 수
        V, D, H = vocab_size, wordvec_size, hidden_size
        rn = np.random.randn

        embed_W = (rn(V, D) / 100).astype('f')
        lstm_Wx = (rn(D, 4 * H) / np.sqrt(D)).astype('f')
        lstm_Wh = (rn(H, 4 * H) / np.sqrt(H)).astype('f')
        lstm_b = np.zeros(4 * H).astype('f')

        # 계층 생성
        self.embed = TimeEmbedding(embed_W)
        self.lstm = TimeLSTM(lstm_Wx, lstm_Wh, lstm_b, stateful=False)

        self.params = self.embed.params + self.lstm.params  # 가중치 매개변수
        self.grads = self.embed.grads + self.lstm.grads # 기울기
        self.hs = None

    def forward(self, xs):
        xs = self.embed.forward(xs)
        hs = self.lstm.forward(xs)
        self.hs = hs
        return hs[:, -1, :] # 마지막 시각의 은닉 상태

    def backward(self, dh):
        dhs = np.zeros_like(self.hs)
         # 마지막 은닉 상태에 대한 기울기 dh를 dhs의 해당 위치에 할당
        dhs[:, -1, :] = dh

        dout = self.lstm.backward(dhs)
        dout = self.embed.backward(dout)
        return dout
```

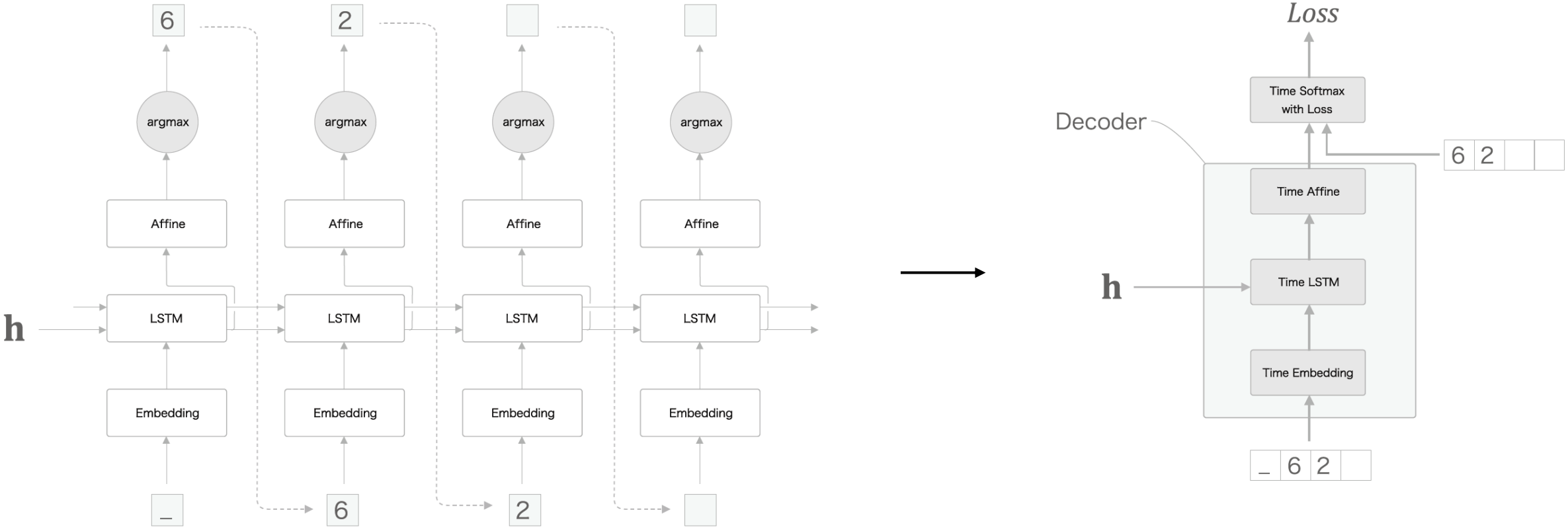
- Decoder



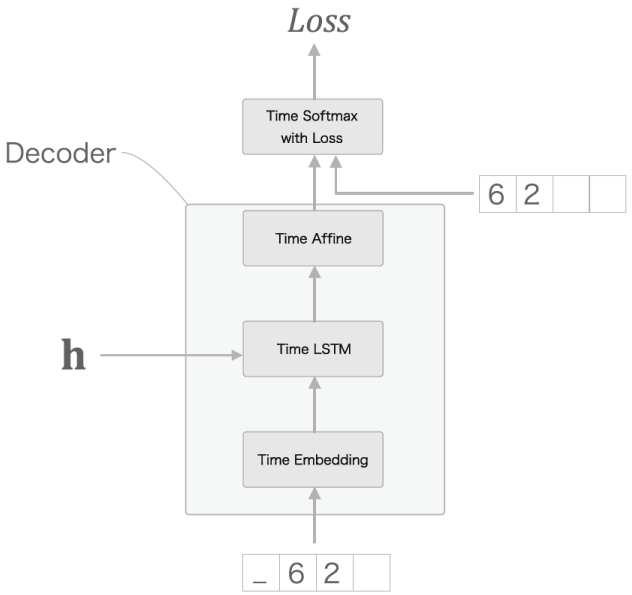


[학습 시 Decoder 계층 구성]

- Decoder의 문장 생성



- Decoder 구현




```python
class Decoder:
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        V, D, H = vocab_size, wordvec_size, hidden_size
        rn = np.random.randn

        embed_W = (rn(V, D) / 100).astype('f')
        lstm_Wx = (rn(D, 4 * H) / np.sqrt(D)).astype('f')
        lstm_Wh = (rn(H, 4 * H) / np.sqrt(H)).astype('f')
        lstm_b = np.zeros(4 * H).astype('f')
        affine_W = (rn(H, V) / np.sqrt(H)).astype('f')
        affine_b = np.zeros(V).astype('f')

        # 계층 생성
        self.embed = TimeEmbedding(embed_W)
        self.lstm = TimeLSTM(lstm_Wx, lstm_Wh, lstm_b, stateful=True)
        self.affine = TimeAffine(affine_W, affine_b)

        self.params, self.grads = [], []
        for layer in (self.embed, self.lstm, self.affine):
            self.params += layer.params
            self.grads += layer.grads

    def forward(self, xs, h):
        self.lstm.set_state(h)

        out = self.embed.forward(xs)
        out = self.lstm.forward(out)
        score = self.affine.forward(out)
        return score

    def backward(self, dscore):
        # dscore : softmax 계층으로부터의 기울기
        dout = self.affine.backward(dscore)
        dout = self.lstm.backward(dout)
        dout = self.embed.backward(dout)
        dh = self.lstm.dh
        return dh
```

```python
def generate(self, h, start_id, sample_size):
  # h: Encoder로부터 받는 은닉 상태
  # start_id: 최초로 주어지는 문자ID
  # sample_size: 생성하는 문자 수
    sampled = []
    sample_id = start_id
    self.lstm.set_state(h)

    for _ in range(sample_size):
        x = np.array(sample_id).reshape((1, 1))
        out = self.embed.forward(x)
        out = self.lstm.forward(out)
        score = self.affine.forward(out)

        sample_id = np.argmax(score.flatten())
        sampled.append(int(sample_id))

    return sampled
```

03 seq2seq 구현

- Seq2seq 클래스 구현

```
class Seq2seq(BaseModel):
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        V, D, H = vocab_size, wordvec_size, hidden_size
        self.encoder = Encoder(V, D, H)
        self.decoder = Decoder(V, D, H)
        self.softmax = TimeSoftmaxWithLoss()

        self.params = self.encoder.params + self.decoder.params
        self.grads = self.encoder.grads + self.decoder.grads

    def forward(self, xs, ts):
        decoder_xs, decoder_ts = ts[:, :-1], ts[:, 1:]

        h = self.encoder.forward(xs)
        score = self.decoder.forward(decoder_xs, h)
        loss = self.softmax.forward(score, decoder_ts)
        return loss

    def backward(self, dout=1):
        dout = self.softmax.backward(dout)
        dh = self.decoder.backward(dout)
        dout = self.encoder.backward(dh)
        return dout

    def generate(self, xs, start_id, sample_size):
        h = self.encoder.forward(xs)
        sampled = self.decoder.generate(h, start_id, sample_size)
        return sampled
```

- Seq2seq 평가

```
import sys
sys.path.append('..')
import numpy as np
import matplotlib.pyplot as plt
from dataset import sequence
from common.optimizer import Adam
from common.trainer import Trainer
from common.util import eval_seq2seq
from seq2seq import Seq2seq
from peeky_seq2seq import PeekySeq2seq


# 데이터셋 읽기
(x_train, t_train), (x_test, t_test) = sequence.load_data('addition.txt')
char_to_id, id_to_char = sequence.get_vocab()

# 하이퍼파라미터 설정
vocab_size = len(char_to_id)
wordvec_size = 16
hideen_size = 128
batch_size = 128
max_epoch = 25
max_grad = 5.0

# 모델/ 옵티마이저/ 트레이너 생성
model = Seq2seq(vocab_size, wordvec_size, hideen_size)
optimizer = Adam()
trainer = Trainer(model, optimizer)

acc_list = []
for epoch in range(max_epoch):
    trainer.fit(x_train, t_train, max_epoch=1,
                batch_size=batch_size, max_grad=max_grad)

    correct_num = 0
    for i in range(len(x_test)):
        question, correct = x_test[[i]], t_test[[i]]
        verbose = i < 10
        correct_num += eval_seq2seq(model, question, correct,
                                    id_to_char, verbose, is_reverse)

    acc = float(correct_num) / len(x_test)
    acc_list.append(acc)
    print('검증 정확도 %.3f%%' % (acc * 100))
```

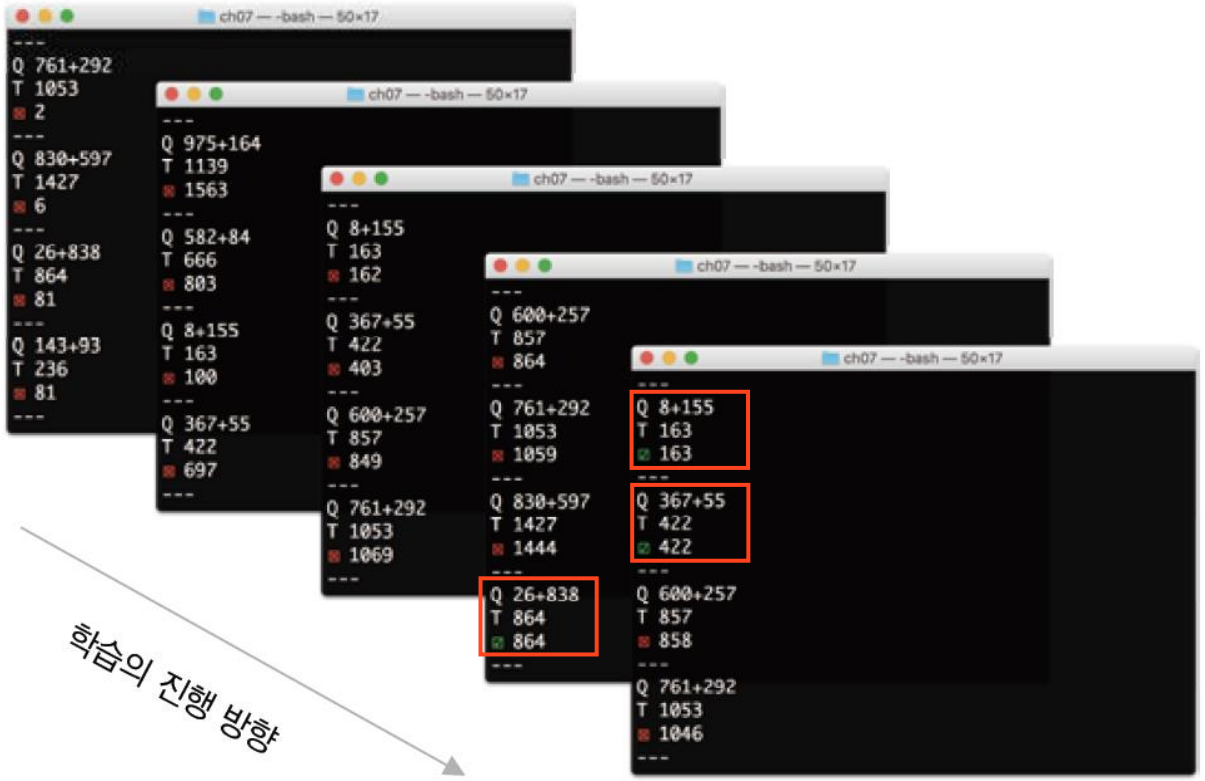
< 학습 과정 >

1. 학습 데이터에서 미니배치 선택
2. 미니배치로부터 기울기 계산
3. 기울기를 사용하여 매개변수 갱신

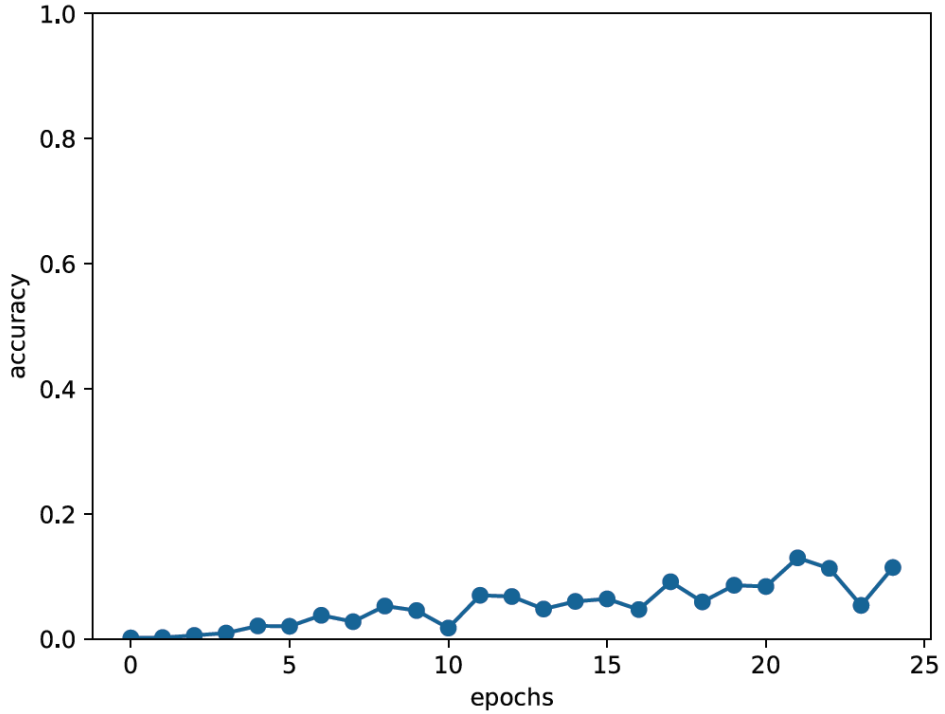
- 평가 척도로 정답률 사용

- eval_seq2seq
  : question을 model에 주고, 문자열을 생성하게 한 후, corret(답)과 같은지를 판단하여 같으면 1, 다르면 0을 반환

- Seq2seq 평가 결과



[정답률 그래프]



-> 에폭이 증가할수록 정답률이 조금씩 상승
-> 최종 정답률 10%

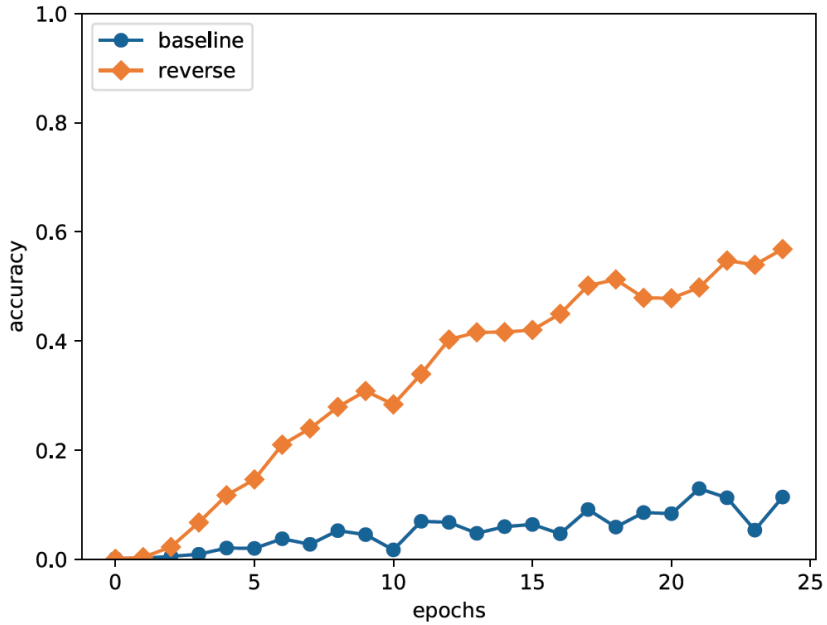
- 입력 데이터 반전 (Reverse)

5	7	+	5			
6	2	8	+	5	2	1
2	2	0	+	8		
⋮						

반전

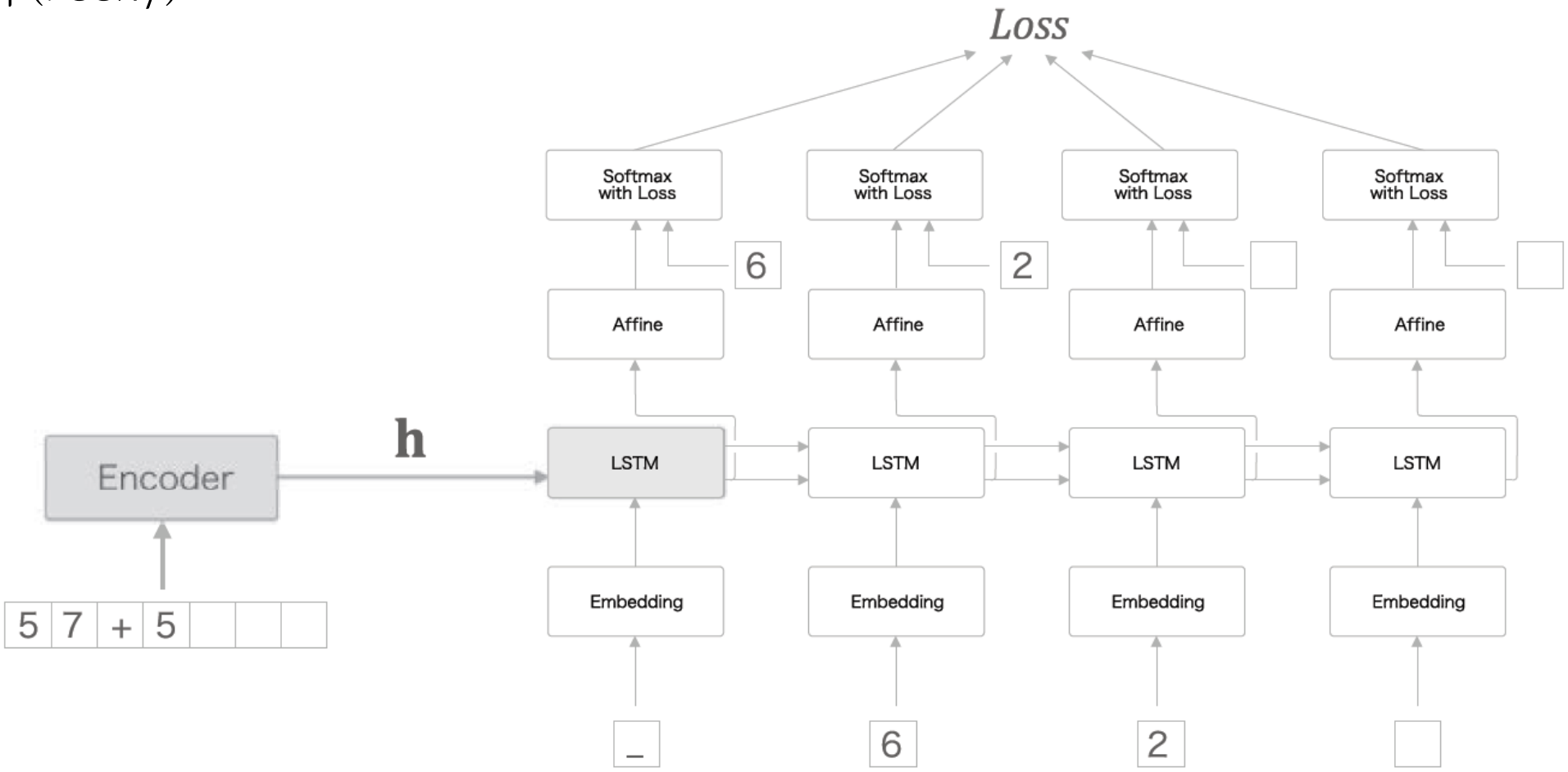
			5	+	7	5
1	2	5	+	8	2	6
		8	+	0	2	2
		⋮				

```
x_train, x_test = x_train[:, ::-1], x_test[:, ::-1]
```



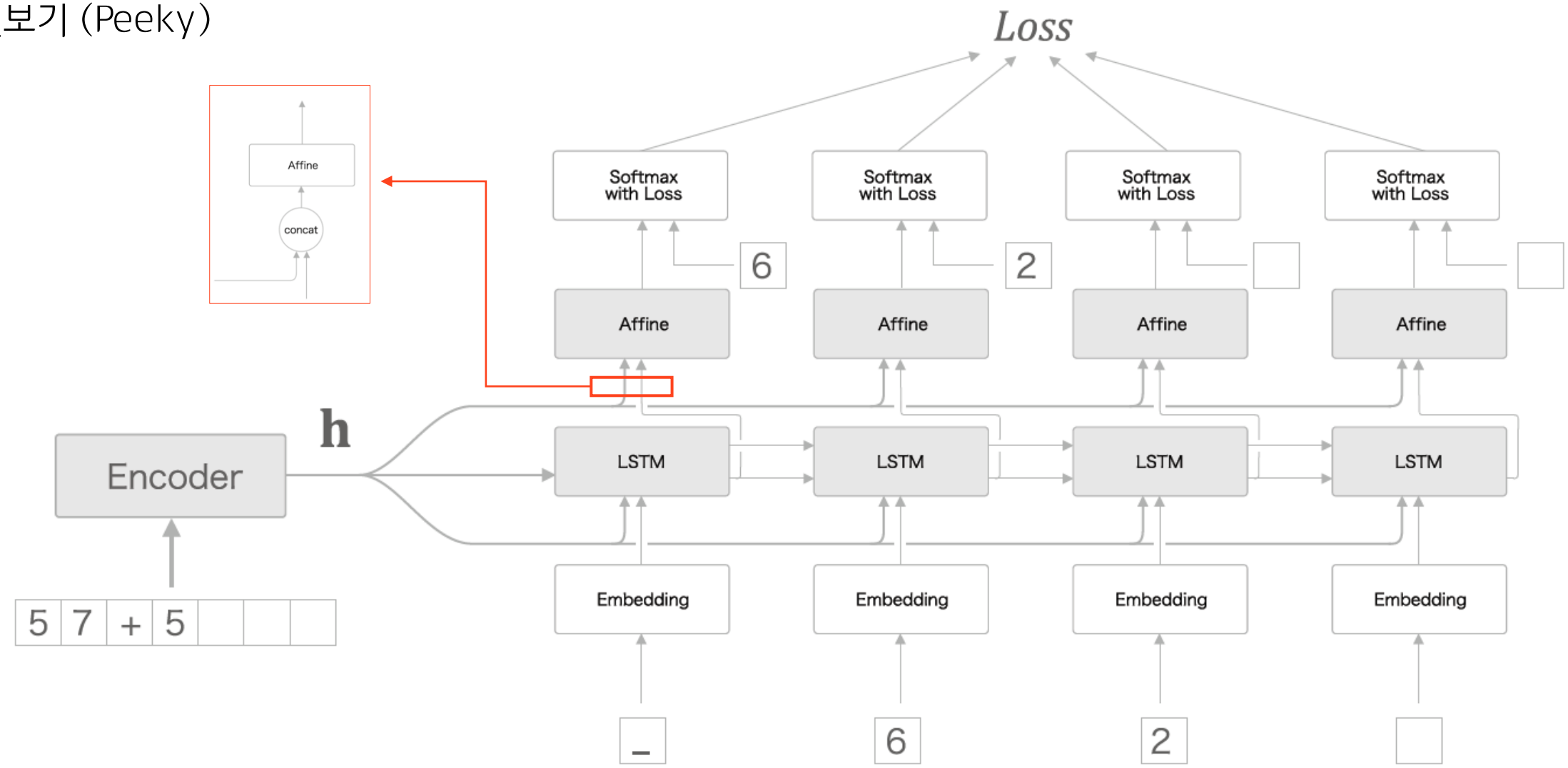
ex) '나는 고양이로소이다' -> 'I am a cat'

- 엿보기 (Peeky)



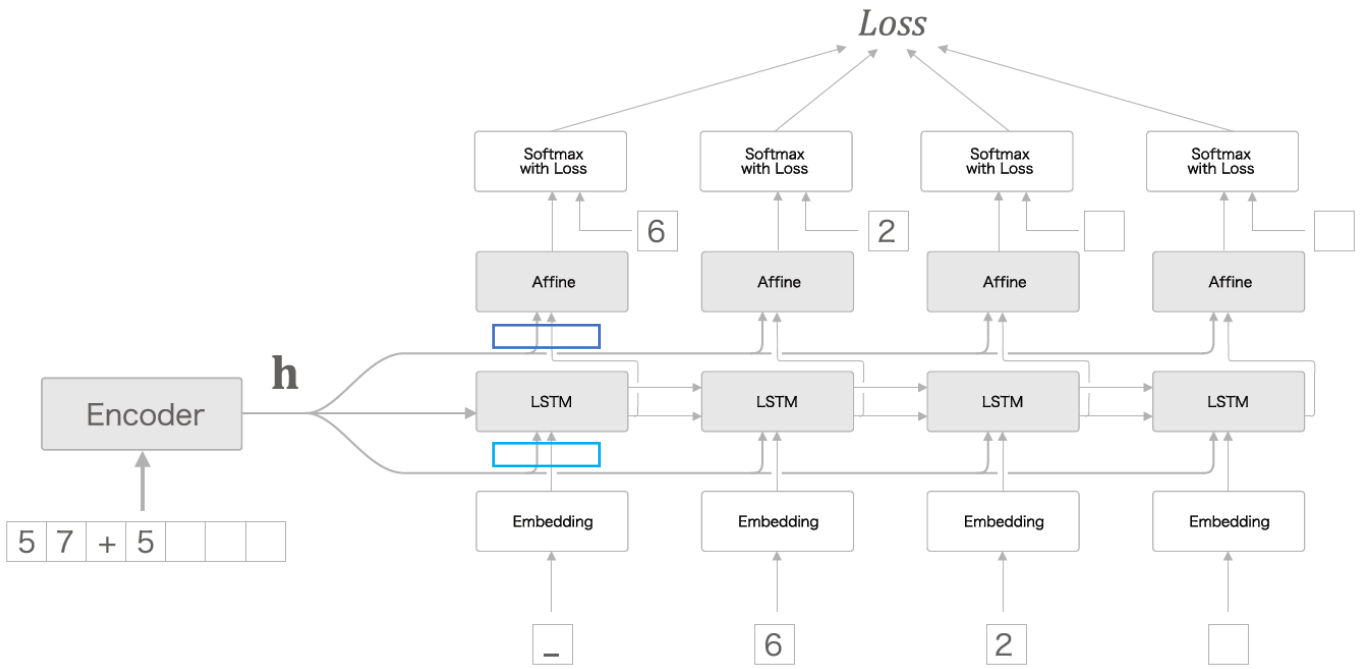
-> 개선 전: Encoder의 출력 h는 첫 번째 LSTM 계층만이 받음

- 엿보기 (Peeky)



-> 개선 후: Encoder의 출력 h를 모든 시각의 LSTM 계층과 Affine 계층에 전해줌

- 엿보기 (Peeky)

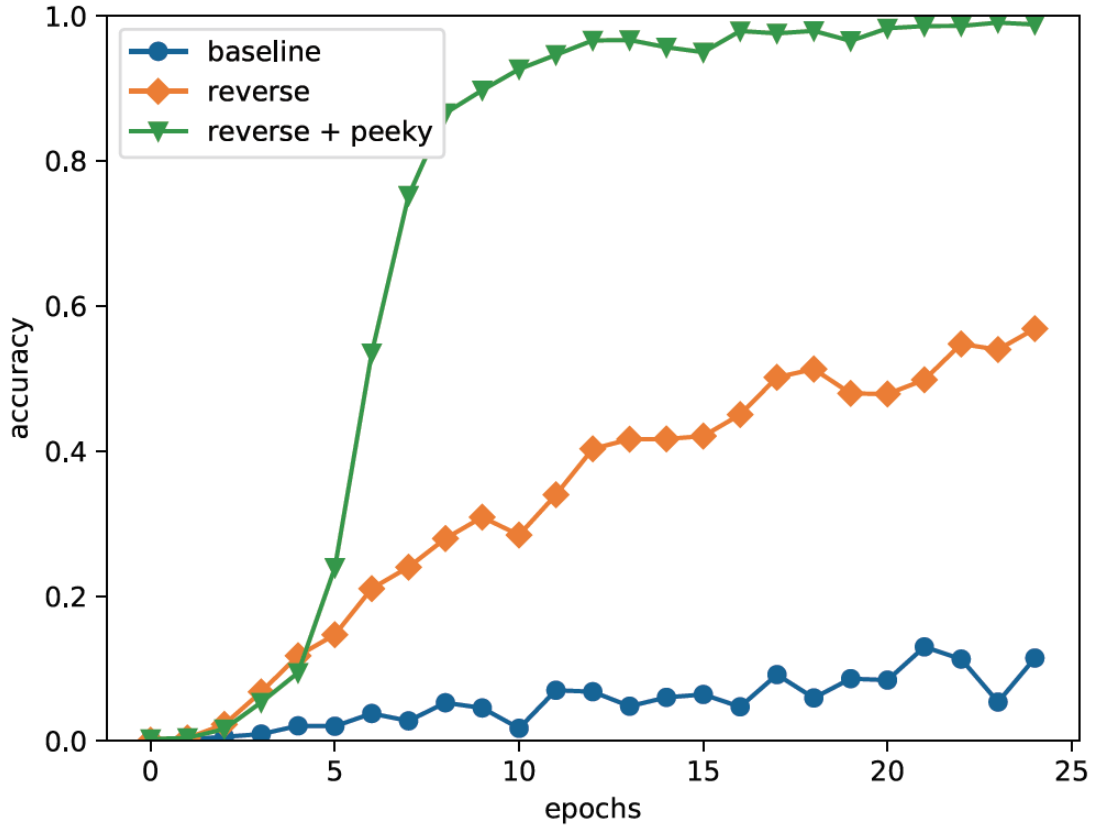



```
class PeekyDecoder:
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        V, D, H = vocab_size, wordvec_size, hidden_size
        rn = np.random.randn

        embed_W = (rn(V, D) / 100).astype('f')
        lstm_Wx = (rn(H + D, 4 * H) / np.sqrt(H + D)).astype('f')
        lstm_Wh = (rn(H, 4 * H) / np.sqrt(H)).astype('f')
        lstm_b = np.zeros(4 * H).astype('f')
        affine_W = (rn(H + H, V) / np.sqrt(H + H)).astype('f')
        affine_b = np.zeros(V).astype('f')

        # 계층 생성
        self.embed = TimeEmbedding(embed_W)
        self.lstm = TimeLSTM(lstm_Wx, lstm_Wh, lstm_b, stateful=True)
        self.affine = TimeAffine(affine_W, affine_b)

        self.params, self.grads = [], []
        for layer in (self.embed, self.lstm, self.affine):
            self.params += layer.params
            self.grads += layer.grads
        self.cache = None

    def forward(self, xs, h):
        N, T = xs.shape
        N, H = h.shape

        self.lstm.set_state(h)

        out = self.embed.forward(xs)
        # h를 시계열만큼 복제해서 hs에 저장
        hs = np.repeat(h, T, axis=0).reshape(N, T, H)
        # hs와 out을 연결
        out = np.concatenate((hs, out), axis=2)
        # lstm 계층에 입력
        out = self.lstm.forward(out)
        # hs와 out을 연결
        out = np.concatenate((hs, out), axis=2)

        score = self.affine.forward(out)
        self.cache = H
        return score
```

- 엿보기 (Peeky)

```
class PeekySeq2seq(Seq2seq):
    def __init__(self, vocab_size, wordvec_size, hidden_size):
        V, D, H = vocab_size, wordvec_size, hidden_size
        self.encoder = Encoder(V, D, H)
        self.decoder = PeekyDecoder(V, D, H)
        self.softmax = TimeSoftmaxWithLoss()

        self.params = self.encoder.params + self.decoder.params
        self.grads = self.encoder.grads + self.decoder.grads
```

```
model = PeekySeq2seq(vocab_size, wordvec_size, hideen_size)
```



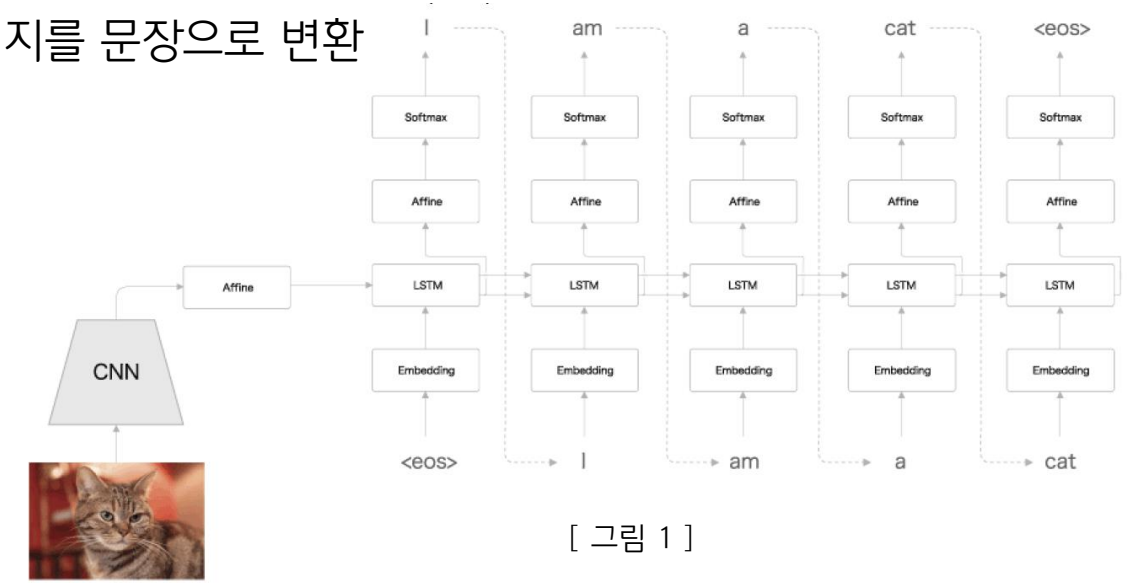
05 seq2seq 사용 예시

- Seq2seq를 이용하는 구체적 예시
  - 기계 번역: ' 한 언어의 문장'을 '다른 언어의 문장 ' 으로 변환
  - 자동 요약: ' 긴 문장'을 '짧게 요약된 문장 ' 으로 변환
  - 질의응답: '질문 ' 을 '응답으로 변환
  - 메일 자동 응답: ' 받은 메일의 문장'을 '답변 글 ' 로 변환

- 챗봇 : 사람과 컴퓨터가 텍스트로 대화를 나누는 프로그램
  - '상대의 말'을 '자신의 말 ' 로 변환하는 문제
  - 대화 기반으로 정답이나 힌트를 얻는 방식은 실용성이 높고 다양하게 응용 가능

- 알고리즘 학습:
  - ex) 파이썬 코드

- 이미지 캡셔닝: 이미지를 문장으로 변환



[그림 1]

A person on a beach flying a kite(해변에서 연 날리는 사람)

A black and white photo of a train on a train track
(선로 위의 열차를 찍은 흑백 사진)

[그림 2]

06 정리

1. RNN을 이용한 언어 모델은 새로운 문장을 생성할 수 있다.
2. 문장을 생성할 때는 하나의 단어(혹은 문자)를 주고 모델의 출력(확률분포)에서 샘플링하는 과정을 반복한다.
3. RNN을 2개 조합함으로써 시계열 데이터를 다른 시계열 데이터로 변환할 수 있다.
4. Seq2seq는 Encoder가 출발어 입력문을 인코딩하고, 인코딩된 정보를 Decoder가 받아 디코딩하여 도착어 출력문을 얻는다.
5. 입력문을 반전시키는 기법(Reverse), 또는 인코딩된 정보를 Decoder의 여러 계층에 전달하는 기법(Peeky)은 seq2seq의 정확도 향상에 효과적이다.
6. 기계 번역, 챗봇, 이미지 캡셔닝 등 seq2seq는 다양한 애플리케이션에 이용할 수 있다.

끝