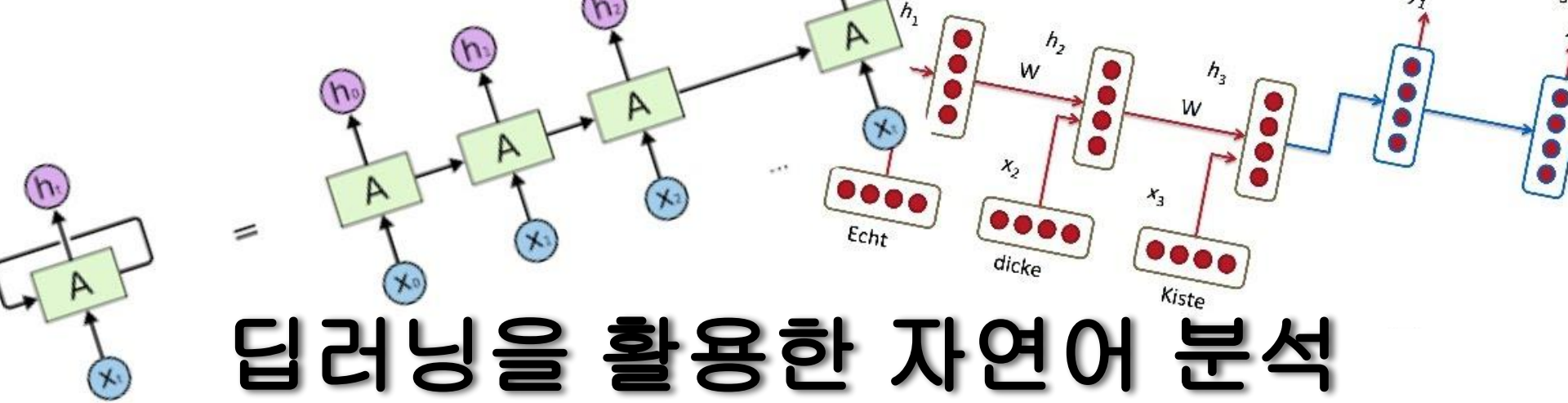
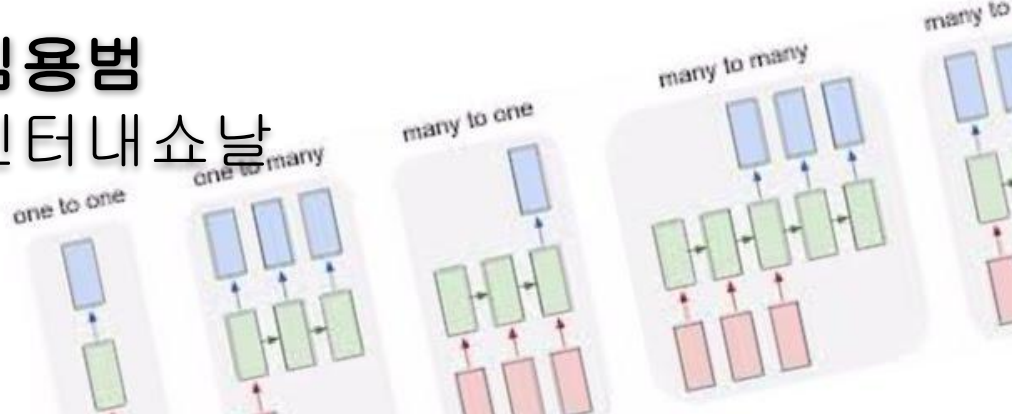


딥러닝을 활용한 자연어 분석

RNN, LSTM, Seq2Seq, Attention

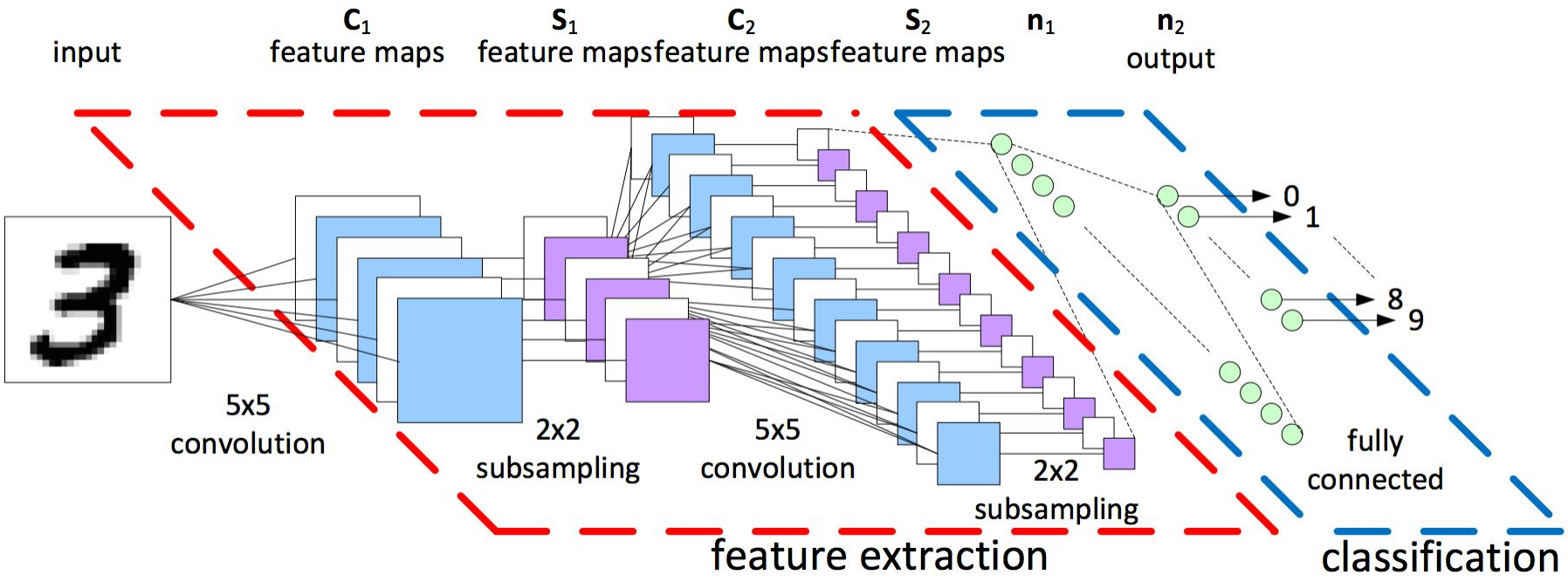
김용범

무영인터내쇼날



CNN

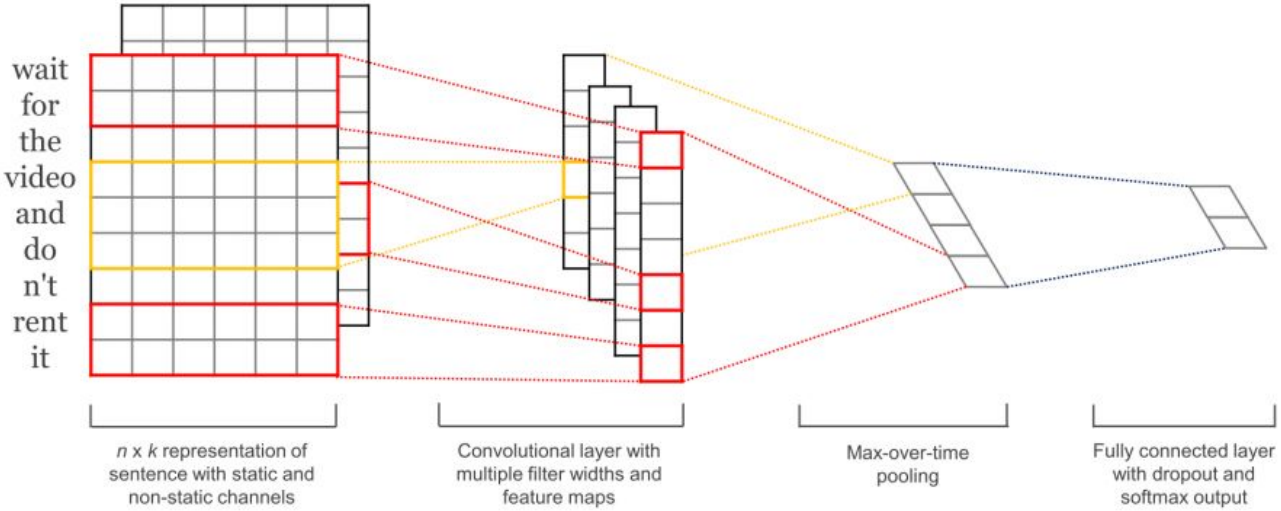
CNN 기본개념


input
C1
feature maps
S1
feature maps
C2
feature maps
S2
feature maps
n1
n2
output
3
5x5
convolution
2x2
subsampling
5x5
convolution
2x2
subsampling
fully
connected
0
1
8
9
feature extraction
classification

RNN

CNN을 활용한 자연어 분석

1. 전방향 합성곱 신경망
2. 방대한 데이터 중 특징적인 부분을 추출하는 기법
3. Text 분류를 CNN을 사용하여 훈련모델을 생성 [Blog]

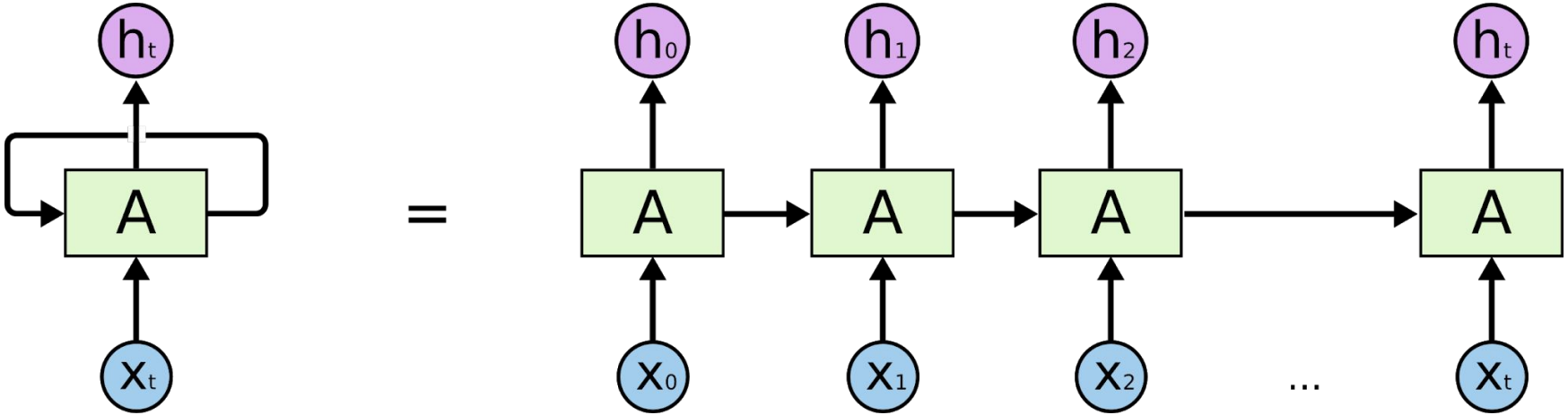


RNN

1. **전방향 신경망** : 회귀모델, 베이지안, CNN
2. **순환 신경망** : Recurrent Neual Network
3. 입력뉴런, 연결뉴런, 출력뉴런등이 연결되어 있다
4. **Gate 반복**에 따라 **각기 다른 결과값**을 Cell위치에서 저장
5. 저장된 다른 위상 값 들을 **평균 또는 합** 등을 구함으로써 압축 가능하다

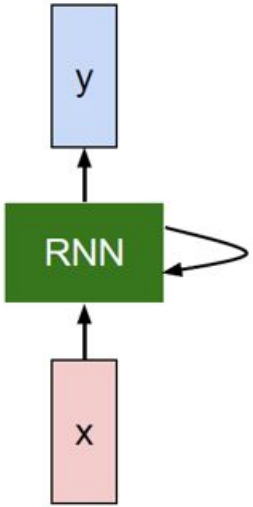
RNN CELL

Recurrent Neual Network

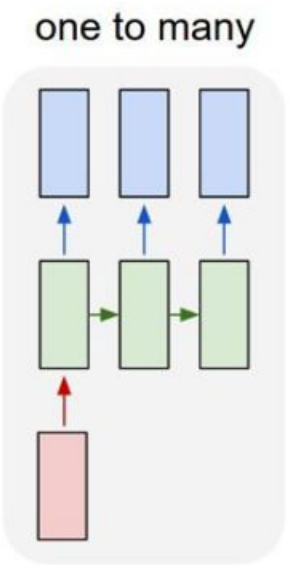


인풋과 **아웃풋**을 모두 받는 **네트워크 구조**로써
다양하고 유연한 구조를 만들 수 있다.

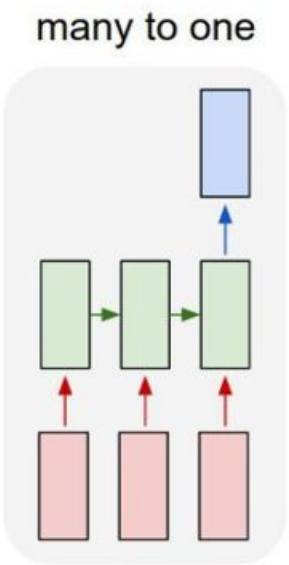
RNN Model



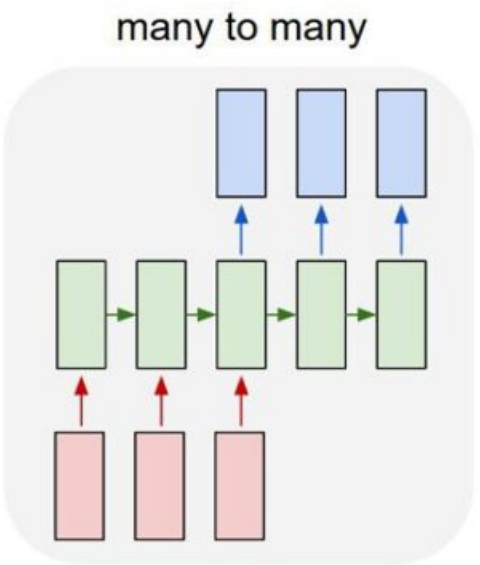
<기본구조>



<사진설명 붙이기>
사진 → 단어들



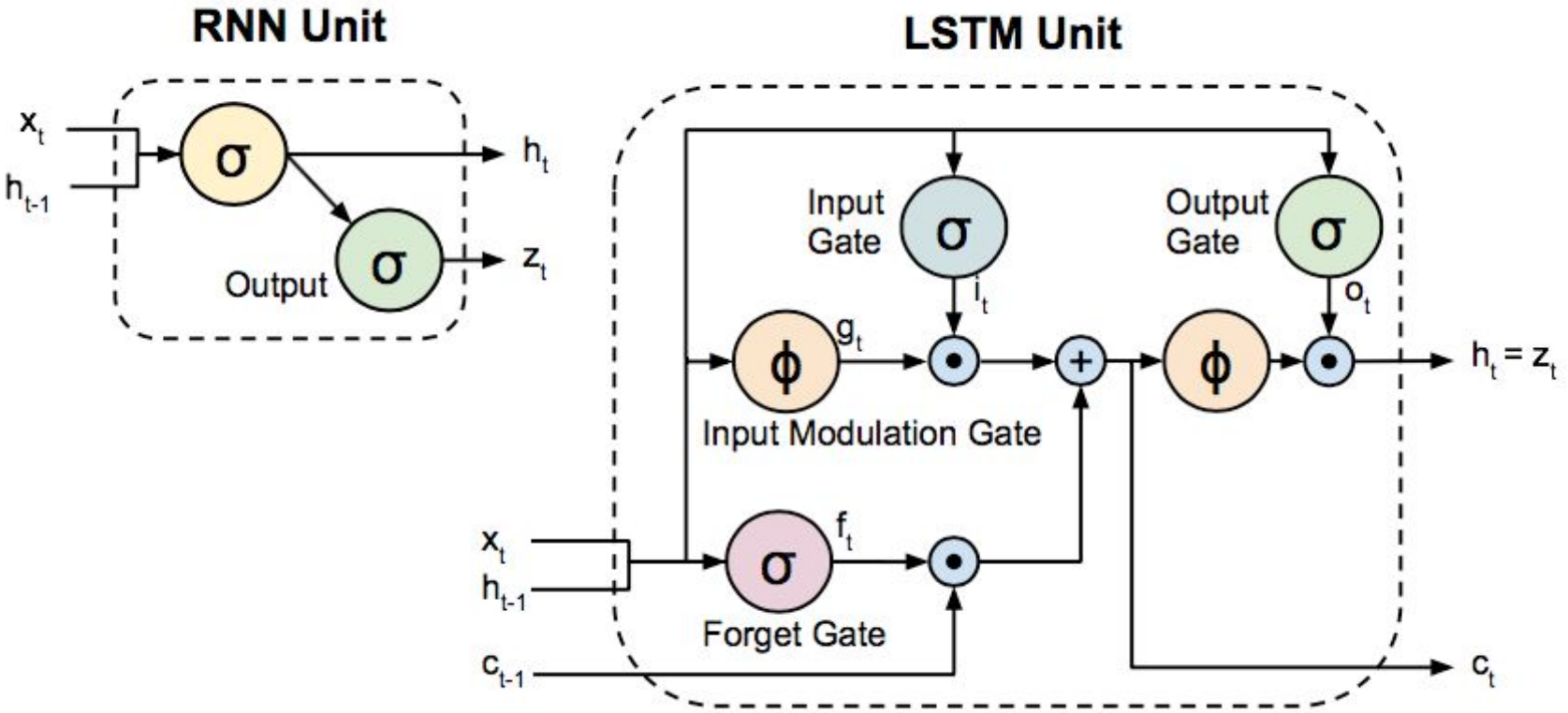
<감성분석>
단어들 → 감성점수



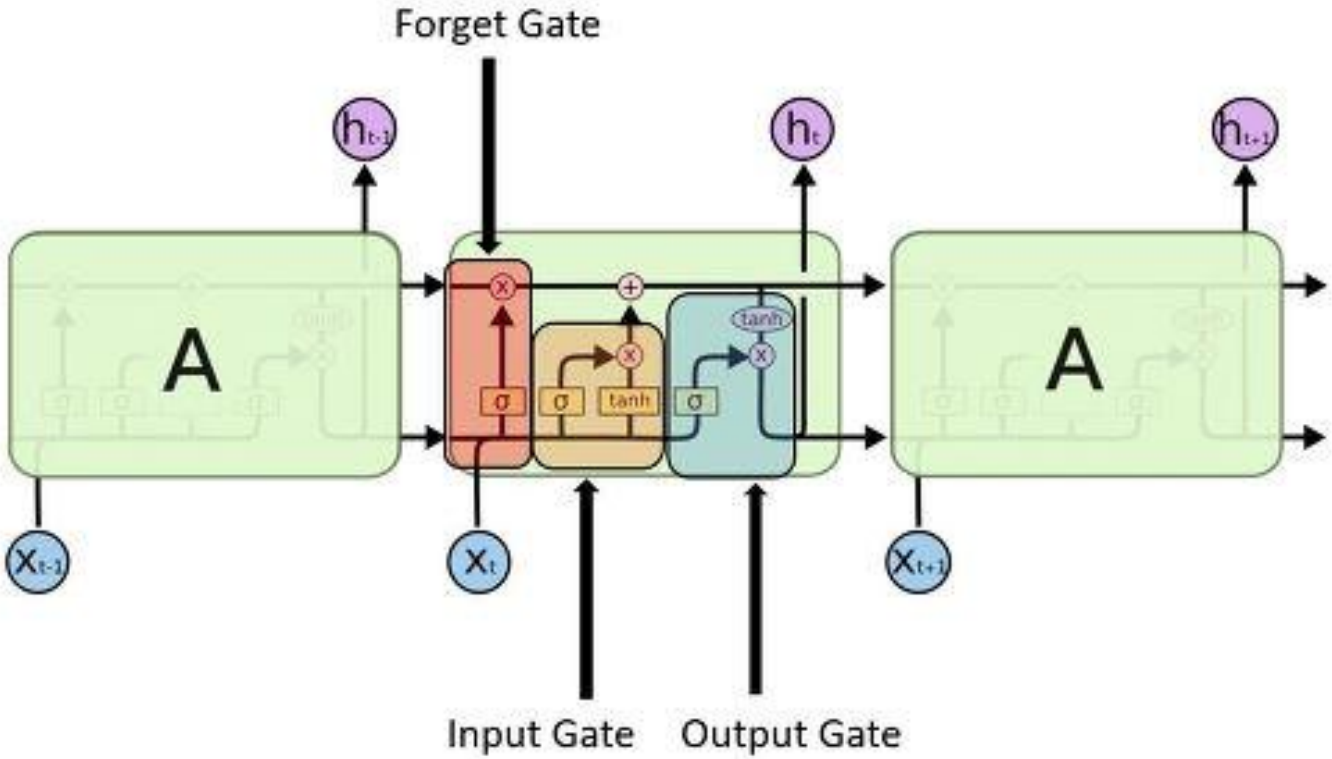
<번역>
단어들 → 단어들

LSTM

LSTM Cell


RNN Unit
x_t
h_{t-1}
σ
h_t
Output
σ
z_t
LSTM Unit
Input Gate
σ
i_t
Output Gate
σ
o_t
g_t
ϕ
Input Modulation Gate
+
ϕ
$h_t = z_t$
x_t
h_{t-1}
σ
f_t
Forget Gate
c_{t-1}
c_t

LSTM Cell

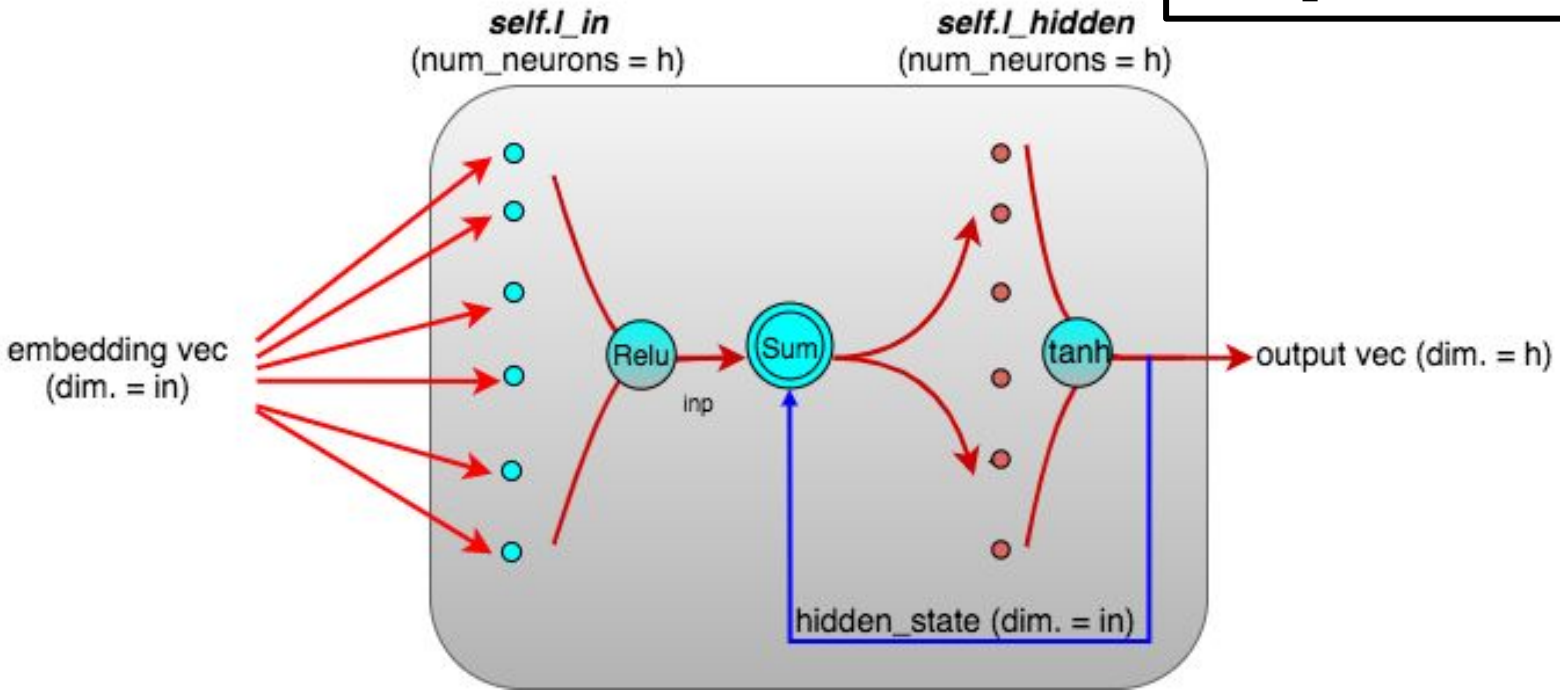


LSTM 유닛구조

1. **유지게이트** : 앞의 **Cell**에서 넘어온 정보 중, 오래된 삭제할 정보와 유지할 정보를 **Sigmoid 뉴런**으로 구분한다
2. **쓰기게이트** : 위에서 구분한 정보중 **필요한 정보(tan h)를 판단** 후 상태변환/유지 여부를 파악 후 **처리**를 한다
3. **출력게이트** : 쓰기게이트와 유사한 구조를 갖고서 **최종 결과물**을 **판단**, **처리**를 한다

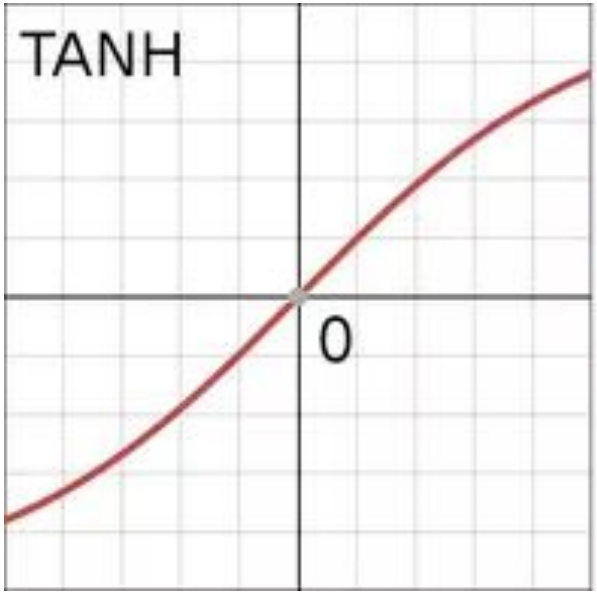
LSTM RNN CELL
Recurrent Neual Network

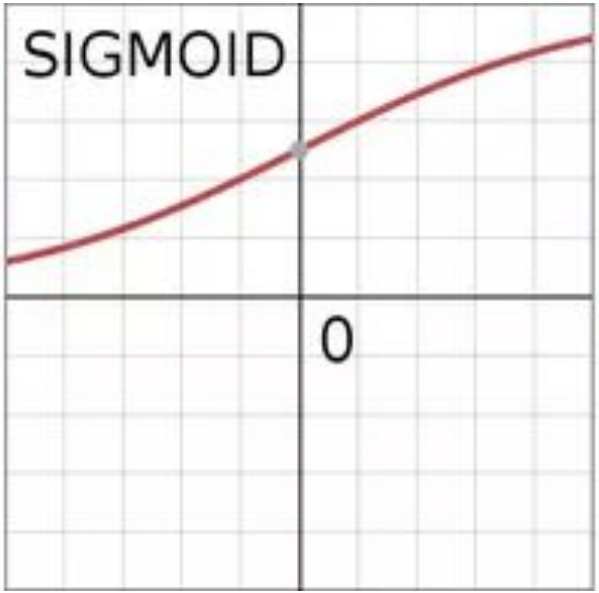
the recurrent connection allows the network **to remember** what it learned in the **previous step.**

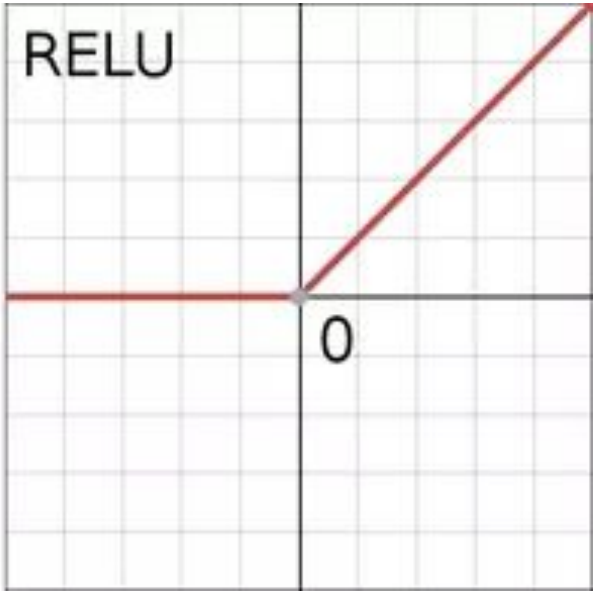


https://medium.com/@apiltamang/unmasking-a-vanilla-rnn-what-lies-beneath-912120f7e56c

활성화 / 게이트 함수들







LSTM 의 장점

1. **Gate**는 공식들 연결 가운데 **Noise**를 발생
2. Noise를 활용하여 **위상차이(반복횟수)**를 구분한다
3. 또는 **Gate 통과 전/이후의 값의 차이**를 활용하는 등 다양한 시간차를 인식 가능하다
4. **Sequence(연속적)** 형태인 원본 데이터를 학습하기 용이하다 (**HMM** 보다 다양한 모델을 구축가능)

[실습Code] - 단어 어휘 예측하기

1. **3개의 알파벳**을 입력하면, **나머지 알파벳 1개**를 정확하게 **예측**하는 모델을 학습한다
2. 알파벳을 기본 요소로써 **LSTM graph**를 활용한다
3. 입력 데이터는 **One hot - Encoding**을 활용
4. One hot-encoding 에 적합한 **별도의 batch 사용자 함수**를 정의한다

LSTM Tensorflow

알파벳 one hot encoding을 정의 후, 학습에 사용할 batch 함수를 정의한다

In [9]:
```
import tensorflow as tf
import numpy as np
char_arr = ['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']
num_dic = {n: i for i, n in enumerate(char_arr)}
dic_len = len(num_dic)
```

In [10]:
```
# 알파벳 배열을 인덱스 번호로 변환 : input_batch, target_batch 에 적용
# [22, 14, 17] [22, 14, 14] [3, 4, 4] [3, 8, 21] ...
def make_batch(seq_data):
    input_batch, target_batch = [], []
    for seq in seq_data:
        input = [num_dic[n] for n in seq[:-1]]
        target = num_dic[seq[-1]]
        input_batch.append(np.eye(dic_len)[input])
        target_batch.append(target)
    return input_batch, target_batch
```

LSTM Tensorflow - LSTM 파라미터, 모델 매개변수를 정의

In [11]:
```
# 1. RNN 신경망 모델 정의
tf.reset_default_graph()
learning_rate         = 0.01
n_hidden, total_epoch = 128, 30
n_step                = 3
n_input = n_class = dic_len
```

In [12]:
```
X = tf.placeholder(tf.float32, [None, n_step, n_input])
Y = tf.placeholder(tf.int32,   [None])
W = tf.Variable(tf.random_normal([n_hidden, n_class]))
b = tf.Variable(tf.random_normal([n_class]))
```

LSTM Tensorflow - LSTM Cell 과 모델 및 비용함수, 활성화 함수를 정의

In [13]:
```
cell1           = tf.nn.rnn_cell.BasicLSTMCell(n_hidden)
cell1           = tf.nn.rnn_cell.DropoutWrapper(cell1, output_keep_prob=0.5)
cell2           = tf.nn.rnn_cell.BasicLSTMCell(n_hidden)
multi_cell      = tf.nn.rnn_cell.MultiRNNCell([cell1, cell2])
outputs, states = tf.nn.dynamic_rnn(multi_cell, X, dtype=tf.float32)
```

In [14]:
```
outputs   = tf.transpose(outputs, [1, 0, 2])
outputs   = outputs[-1]
model     = tf.matmul(outputs, W) + b
cost      = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
                           logits = model, labels = Y))
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

LSTM Tensorflow - graph 를 학습한다

In [15]:
```
# Train 데이터를 정의한다
seq_data = ['word', 'wood', 'deep', 'dive', 'cold', 'cool', 'load', 'love', 'kiss', 'kind']
```

In [16]:
```
# 2. RNN 신경망 학습
sess = tf.Session()
sess.run(tf.global_variables_initializer())
input_batch, target_batch = make_batch(seq_data)
for epoch in range(total_epoch):
    _, loss = sess.run([optimizer, cost],
                       feed_dict={X: input_batch, Y: target_batch})
    if epoch % 3 == 0:
        print('Epoch: {:.4f} cost = {:.6f}'.format(epoch + 1, loss))
print('최적화 완료!')
```

```
Epoch: 1.0000 cost = 3.891259
Epoch: 4.0000 cost = 1.366117
Epoch: 7.0000 cost = 0.578614
Epoch: 10.0000 cost = 0.278150
Epoch: 13.0000 cost = 0.360198
Epoch: 16.0000 cost = 0.123838
Epoch: 19.0000 cost = 0.094416
Epoch: 22.0000 cost = 0.036910
Epoch: 25.0000 cost = 0.046367
Epoch: 28.0000 cost = 0.145751
최적화 완료!
```

LSTM Tensorflow - 학습한 모델을 평가한다

In [17]:
```
# 3. 모델의 성능을 평가
prediction       = tf.cast(tf.argmax(model, 1), tf.int32)
prediction_check = tf.equal(prediction, Y)
accuracy         = tf.reduce_mean(tf.cast(prediction_check, tf.float32))
input_batch, target_batch = make_batch(seq_data)
predict, accuracy_val     = sess.run([prediction, accuracy],
                                     feed_dict={X: input_batch, Y: target_batch})
```

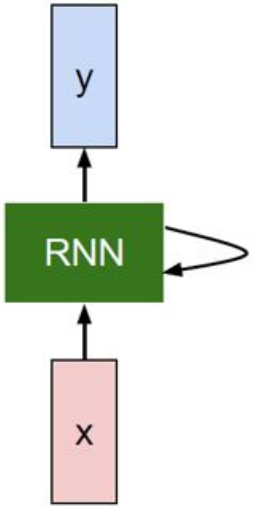
In [18]:
```
predict_words = []
for idx, val in enumerate(seq_data):
    last_char = char_arr[predict[idx]]
    predict_words.append(val[:3] + last_char)

print('\n=== 예측 결과 ===')
print('입력값:', [w[:3] + ' ' for w in seq_data])
print('예측값:', predict_words)
print('정확도:', accuracy_val)
sess.close()
```

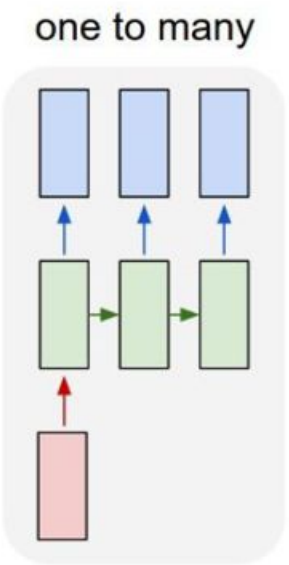
```
=== 예측 결과 ===
입력값: ['wor ', 'woo ', 'dee ', 'div ', 'col ', 'coo ', 'loa ', 'lov ', 'kis ', 'kin ']
예측값: ['word', 'wood', 'deep', 'dive', 'cold', 'cool', 'load', 'love', 'kiss', 'kind']
정확도: 1.0
```

Seq2Seq

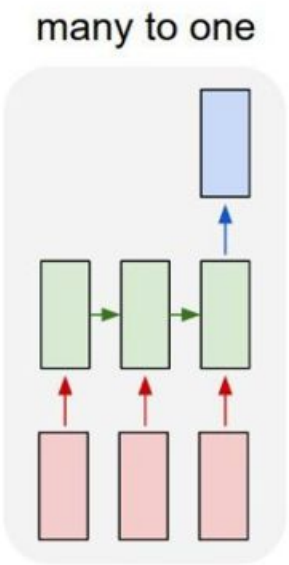
RNN Model



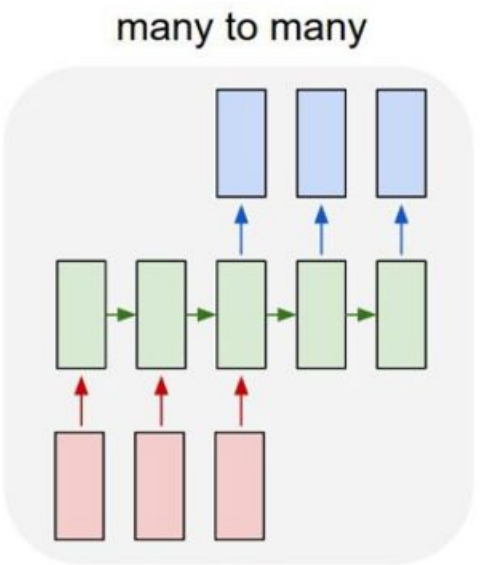
<기본구조>



<사진설명 붙이기>
사진 → 단어들

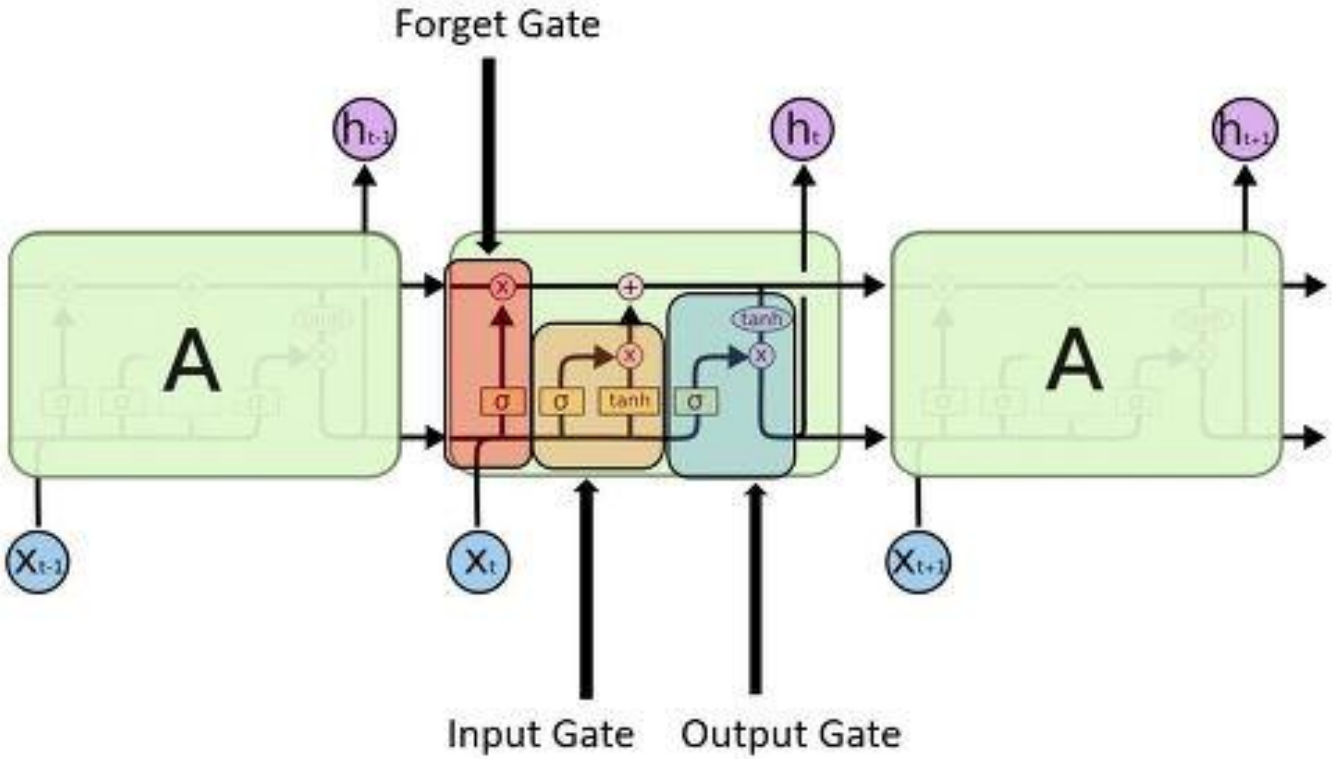


<감성분석>
단어들 → 감성점수

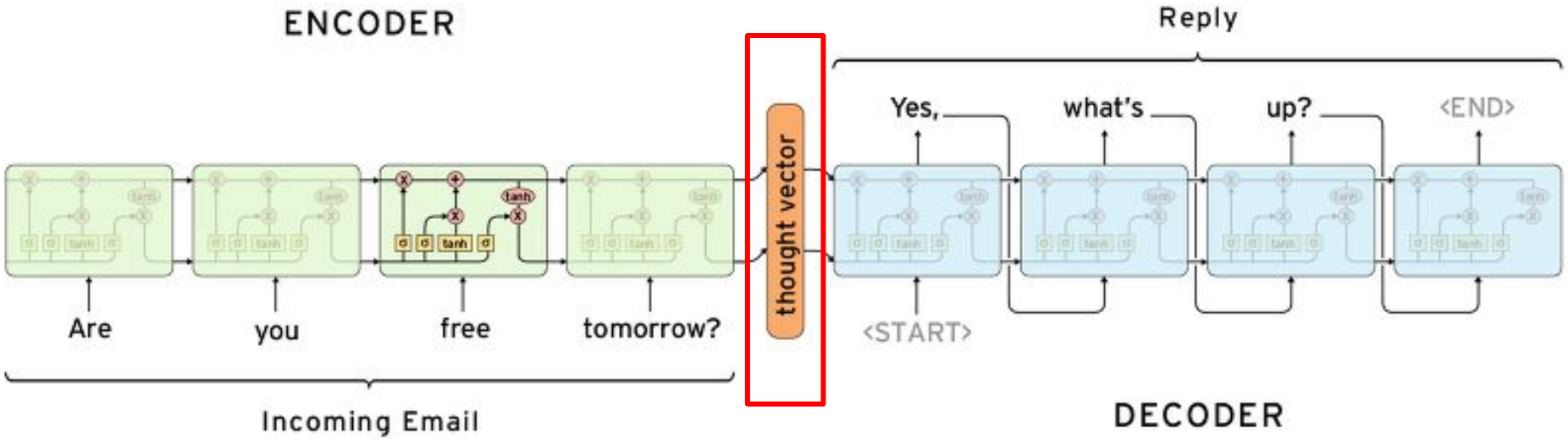


<번역>
단어들 → 단어들

LSTM Cell



seq 2 seq Cell



seq 2 seq

1. 2013년 **구글**에서 공개한 **기계번역** 신경망 모델이다
2. **LSTM 인코더**(A, B, C) 와 **LSTM 디코더**(W, X, Y,)를 연결
3. **RNN의 재귀적 학습**의 특성으로 인해, **특수함 심벌** ((1) **입력의 시작**을 알림 (2) 디코더 **출력이 끝남**을 알림)을 필요로 한다
4. **대화형 챗봇**, **번역** 등 폭넓게 활용

seq 2 seq

1. CNN 의 모델중 하나인 **GAN**과 유사한 구조를 갖는다
2. **GAN**은 **전방향 학습망** 으로써, 입력 이미지와 출력 이미지 중간과정이 별도로 관리가 되지 않아 **학습할 때마다 다른 결과를 출력**한다
3. 반면 **seq2seq** 는 보다 **일관성 있는 결과물**을 출력한다

[실습Code] - 번역봇

1. **영문**과 이에 대응하는 **한글**을 학습
2. 객체들은 **One-Hot Encoding**을 활용
3. **한글** 과 **영문의 갯수**는 정교한 학습을 위해 동일하게 한다
4. 글자수가 다른 내용을 학습 할 경우는 **Padding 기호**를 활용

seq2seq Tensorflow - 학습에 사용할 영문과 한글을 정의한다

In [1]:
```
import tensorflow as tf
import numpy as np

char_arr = [ c for c in 'SPabcdefghijklmnopqrstuvwxyz단어나무놀이소녀키스사랑E']
num_dic  = { n : i for i, n in enumerate(char_arr)}
dic_len  = len(num_dic)
```

In [2]:
```
seq_data = [['word', '단어'], ['wood', '나무'], ['game', '놀이'],
            ['girl', '소녀'], ['kiss', '키스'], ['love', '사랑']]

def make_batch(seq_data):
    input_batch, output_batch, target_batch = [], [], []
    for seq in seq_data:
        input_txt  = [num_dic[n] for n in seq[0]]
        output_txt = [num_dic[n] for n in ('S' + seq[1])]
        target     = [num_dic[n] for n in (seq[1] + 'E')]
        input_batch.append(np.eye(dic_len)[input_txt])
        output_batch.append(np.eye(dic_len)[output_txt])
        target_batch.append(target)
    return input_batch, output_batch, target_batch
```

seq2seq Tensorflow - encoder 와 decoder 를 정의한다

In [15]:
```
# 2. 모델의 정의
tf.reset_default_graph()
learning_rate         = 0.01
n_hidden, total_epoch = 128, 100
n_class = n_input = dic_len

enc_input = tf.placeholder(tf.float32, [None, None, n_input])
dec_input = tf.placeholder(tf.float32, [None, None, n_input])
targets   = tf.placeholder(tf.int64, [None, None])
```

In [16]:
```
with tf.variable_scope('encode'):
    enc_cell = tf.nn.rnn_cell.BasicRNNCell(n_hidden)
    enc_cell = tf.nn.rnn_cell.DropoutWrapper(enc_cell, output_keep_prob = 0.5)
    outputs, enc_states = tf.nn.dynamic_rnn(enc_cell, enc_input, dtype=tf.float32)
```

In [17]:
```
with tf.variable_scope('decode'):
    dec_cell = tf.nn.rnn_cell.BasicRNNCell(n_hidden)
    dec_cell = tf.nn.rnn_cell.DropoutWrapper(dec_cell, output_keep_prob=0.5)
    outputs, dec_states = tf.nn.dynamic_rnn(dec_cell, dec_input,
                                            initial_state = enc_states,
                                            dtype = tf.float32)
```

In [18]:
```
model     = tf.layers.dense(outputs, n_class, activation=None)
cost      = tf.reduce_mean(tf.nn.sparse_softmax_cross_entropy_with_logits(
                                logits = model, labels = targets))
optimizer = tf.train.AdamOptimizer(learning_rate).minimize(cost)
```

seq2seq Tensorflow - 모델을 학습한다

In [19]:
```
# 3. 모델의 학습
sess = tf.Session()
sess.run(tf.global_variables_initializer())
input_batch, output_batch, target_batch = make_batch(seq_data)
for epoch in range(total_epoch):
    _, loss = sess.run([optimizer, cost],
                       feed_dict={enc_input: input_batch,
                                  dec_input: output_batch,
                                  targets: target_batch})
    if epoch % 5 == 0 :
        print('Epoch: {:4d}  cost = {:.6f}'.format((epoch + 1),loss))
print('최적화 완료!')
```

```
Epoch:    1  cost = 3.623010
Epoch:    6  cost = 0.315577
Epoch:   11  cost = 0.060861
Epoch:   16  cost = 0.101746
Epoch:   21  cost = 0.021950
Epoch:   26  cost = 0.014644
Epoch:   31  cost = 0.001413
Epoch:   36  cost = 0.001478
Epoch:   41  cost = 0.004127
Epoch:   46  cost = 0.001251
```

seq2seq Tensorflow - 학습 모델로 단어를 예측해본다

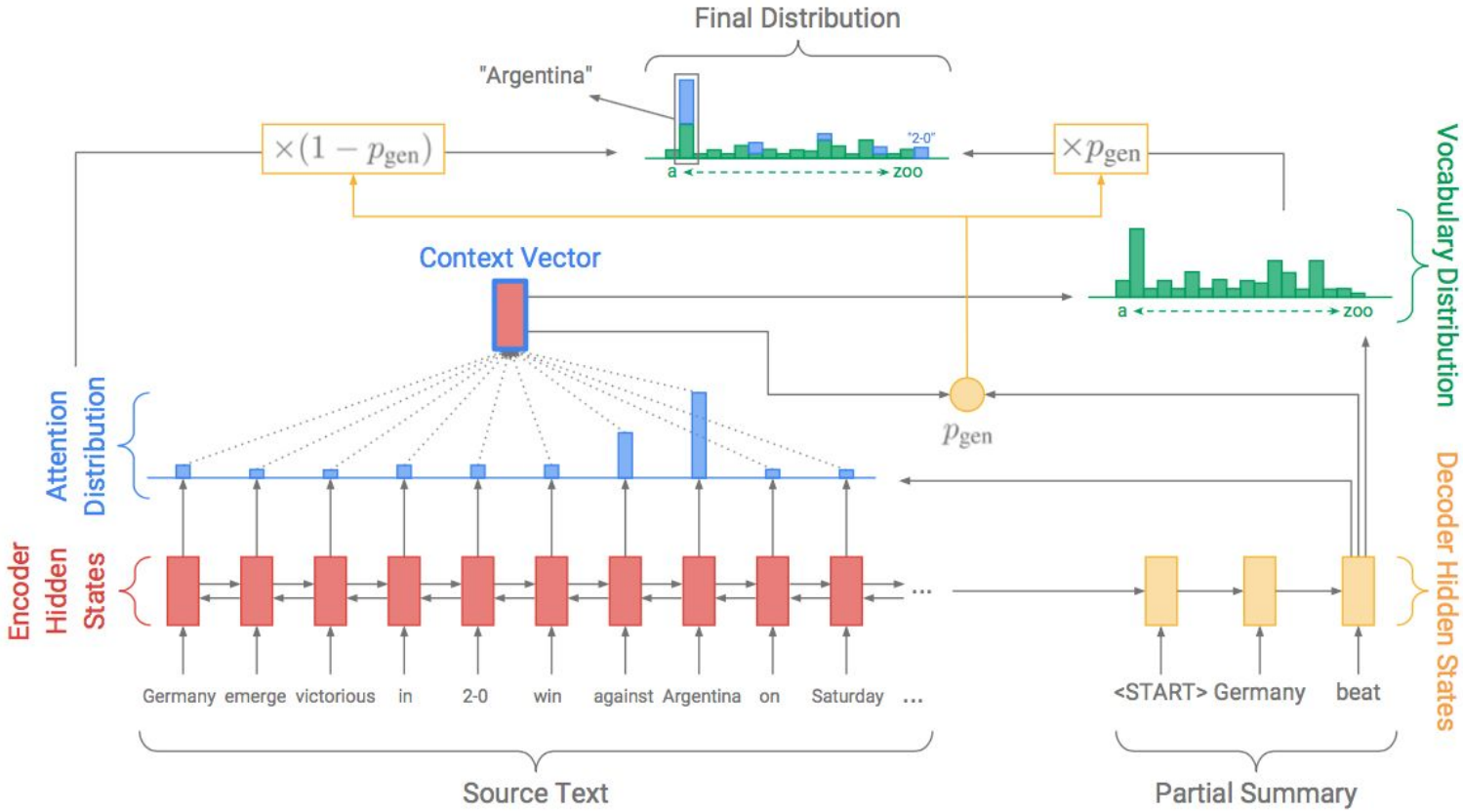
In [19]:
```
def translate(word):
    seq_data = [word, 'P' * len(word)]
    input_batch, output_batch, target_batch = make_batch([seq_data])
    prediction = tf.argmax(model, 2)    # [None, None, n_input]
    result = sess.run(prediction,
                      feed_dict={enc_input: input_batch,
                                 dec_input: output_batch,
                                 targets: target_batch})
    decoded = [char_arr[i] for i in result[0]]
    end = decoded.index('E')
    translated = ''.join(decoded[:end])
    return translated

print('\n=== 번역 테스트 ===')
print('word ->', translate('word'))
print('wodr ->', translate('wodr'))
print('love ->', translate('love'))
print('loev ->', translate('loev'))
print('abcd ->', translate('abcd'))
sess.close()
```

```
=== 번역 테스트 ===
word -> 단어
wodr -> 단무
love -> 사랑
loev -> 사랑
abcd -> 키스단무
```

ATTENTION

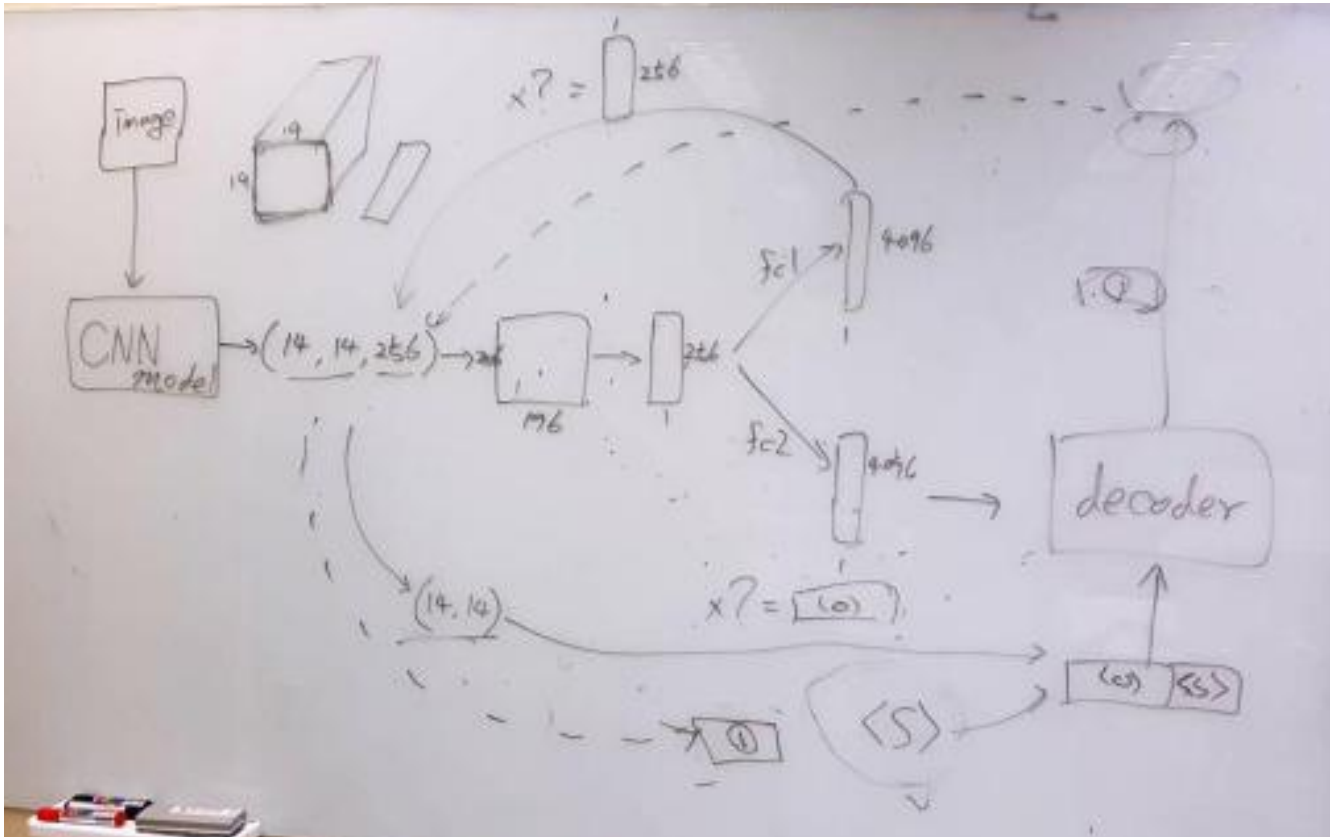
Attention 모델 - 뉴스 제목 추출하기



Attention 내용

1. seq2seq 에서 **LSTM의 연속적인 분석능력**을 Encoder 와 Decoder 로 연결하여 **2배의 길이**의 성능을 활용
2. 하지만 길이가 4, 5이 아닌 **20개 30개의 LSTM을 seq2seq** 로 만든다면 그 성능또한 한계가 존재
3. LSTM 셀들의 **가중치를 별도로 학습하는 보조 Cell**을 추가하여 다양한 길이의 객체도 학습가능한 모델을 제안

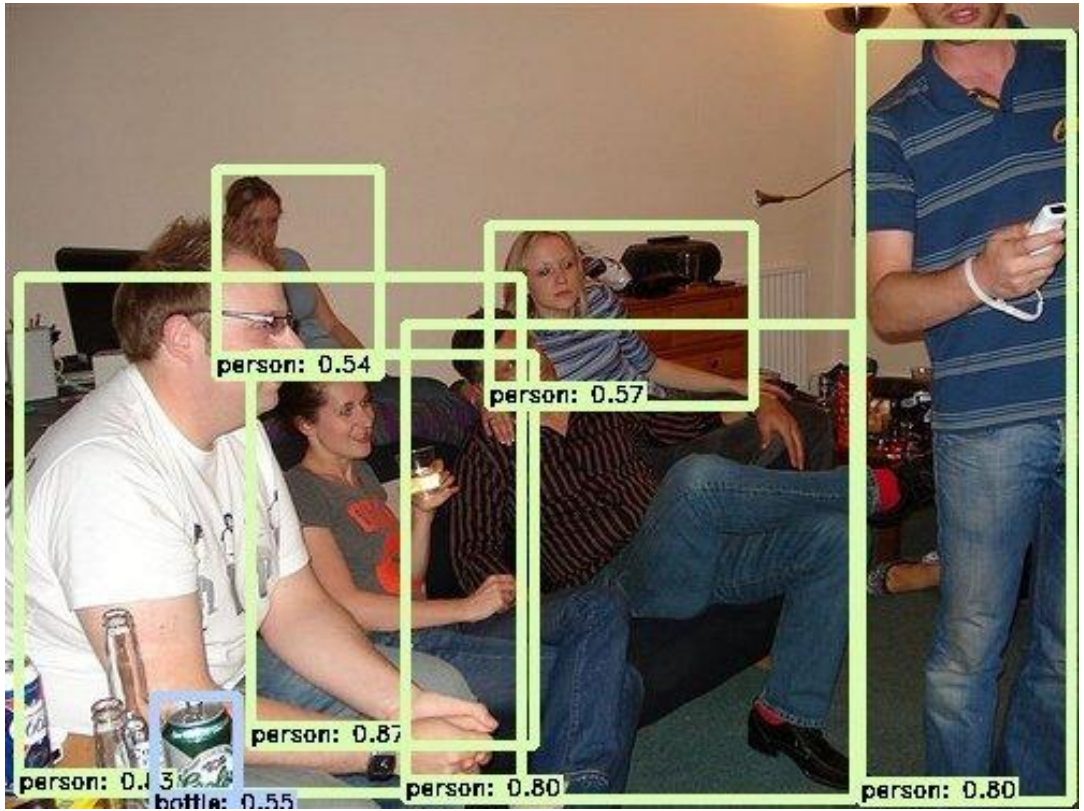
Attention with CNN & LSTM



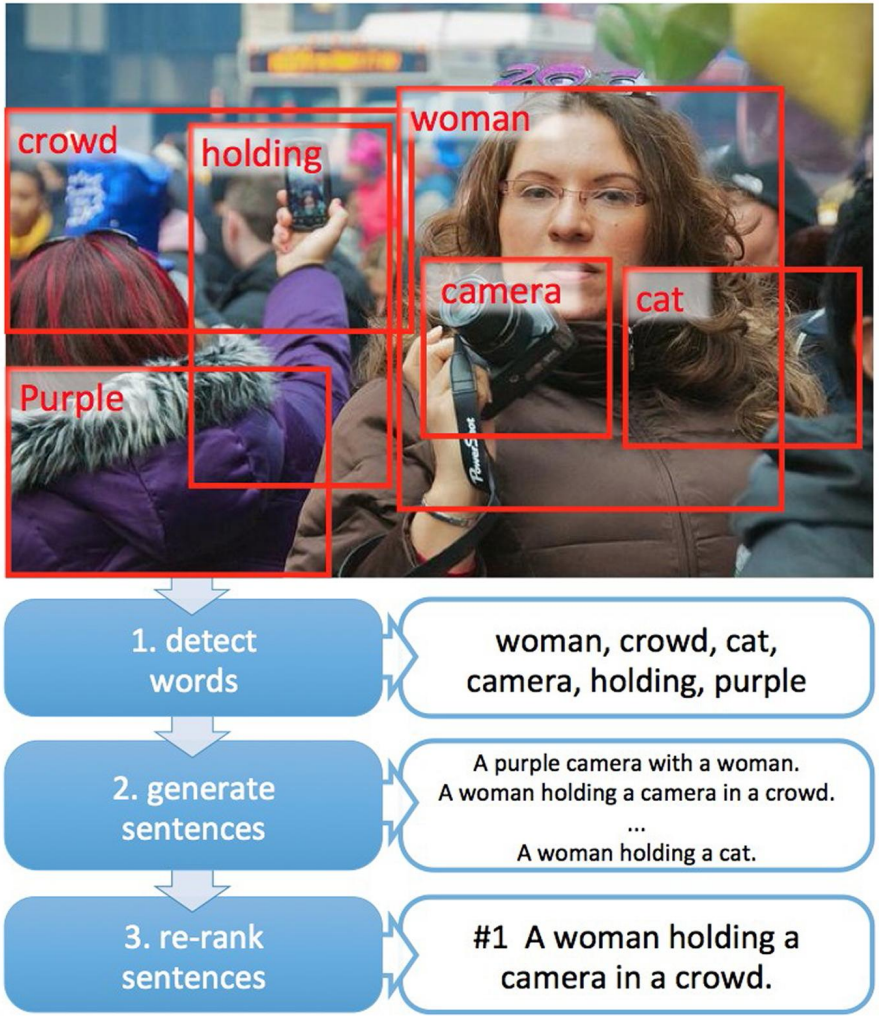
Attention with CNN & LSTM

https://www.oreilly.com/ideas/interpretability-via-attentional-and-memory-based-interfaces-using-tensorflow

Image Detection



Attention and **LSTM**



마치며

번역기 자막의 수준

인간번역 자막의 논란

청와대 | 문재인 대통령 | 청와대 뉴스룸 | 정책자료 | 국민소통 광장 | 청와대 알림 | 청와대 관람 신청

청원진행중

박지훈 번역가의 작품(번역)참여를 반대합니다

참여인원 : [3,176명]

카테고리 문화/예술/체육/언론 **청원시작** 2018-04-25 **청원마감** 2018-05-25 **청원인** naver - ***

청원시작 — **청원진행중** — 청원종료 — 브리핑

청원개요

지금까지 영화에서 수많은 오역과 발번역을 하여 각종 비난과 퇴출운동이 일어나고 있는
문제의 박지훈 번역가의 작품(번역)참여를 반대하고 퇴출을 원합니다

이 번역가의 영어 실력이 기본도 안돼있다는 사실을
기초회화도 불가능한 수준의 영어실력을 가진 일반일들도 인지할 수 있는 정도입니다

오늘의 청와대 LIVE

추천순 TOP 5

1. 다산신도시 실버랙배 비용은 입주민들의 관리비로…
2. 선관위의 위법사항 내용에 따른 국회의원 전원 위…
3. 삼성증권 시스템 규제와 공매도 금지
4. 티비 조선의 종편 허가 취소 청원
5. 세월호 관련 청문회 위증한 조여옥대위 징계바랍…

언어의 유형

1. **분석어 = 고립어** : 중국어, 동남아 계통. 변형 없이, 단어가 깔끔하게 분리(분석)
2. **종합어 = 포합어, 굴절어, 교착어** : 단어 변형 있다.
3. **교착어 = 첨가어** : 한국어, 일본어계통. 어간 어미 구분, 어미가 변함
4. **굴절어 = 융합어** : 유럽 ~ 아랍 ~ 인도. 인도유럽어족(백인계통). 어간 어미 구분 없이 단어가 변함
5. **포합어** : 인디언 계통, 여러 단어가 변화하여 섞이므로 단어 구분이 어렵다.
6. 단순한 정도 : (가장 복잡)포합→첨가→굴절→고립(가장 단순)

Deep Learning 과 자연어

1. 3행시, 끝말 잇기, 말투 따라하기, 단문분석
2. **이미지, 음성** 등 다른데이터와 **자연어의 차이점**
3. 단계 세분화, **Rule Based(통계)**를 활용한 **안정성** 추구
4. **머신러닝은 보조재**로써 활용

Q/A

수고하셨습니다