PyCon Korea 2019

딥러닝 NLP 손쉽게 따라해보기
- GluonNLP-

RNN, Attention, BERT with GluonNLP

순서

1. RNN
2. Attention
3. Language Model
4. BERT

시퀀스 데이터

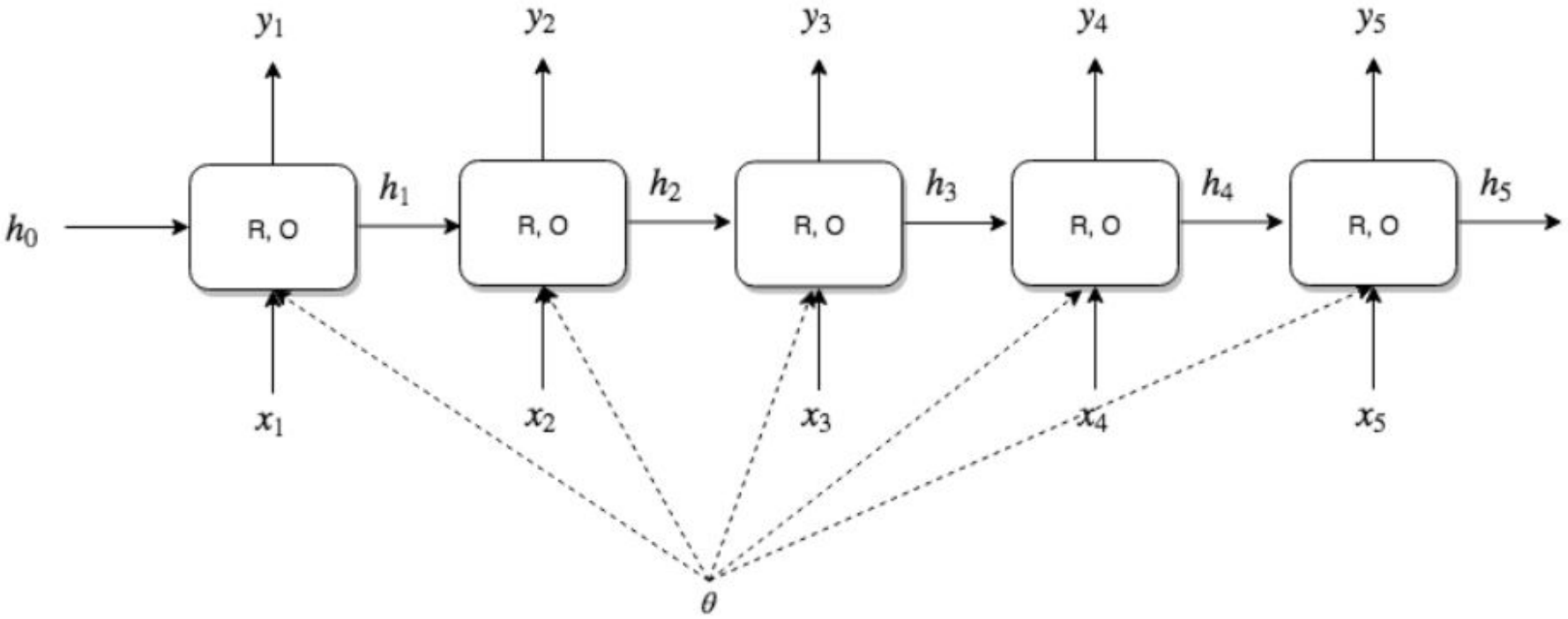
- 순서의 중요성
  - 혹평 그리고 더 많았던 찬사 vs 찬사 그리고 더 많았던 혹평
- 다양한 문장, 문서 길이
- 문맥의 중요성
  - **주말**엔 좀더 늦게 일어나고 아침과 점심을 간단하게 하기도 한다.
  - 장기 문맥 의존성(long-term context dependency)

RNN의 구조

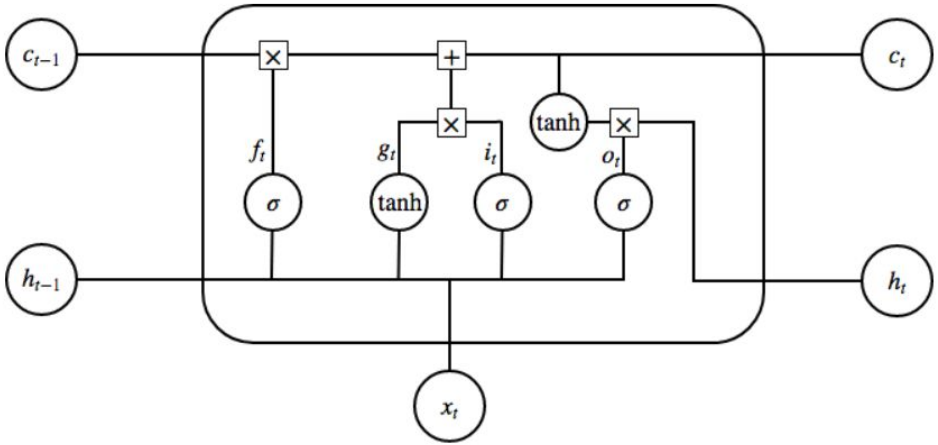
- 읽기
  - 이전 단어를 기억하고 다음 단어를 읽을때 활용하는 과정을 반복
- RNN
  - 이러한 인지적 과정을 도입해 만든 네트워크 구조

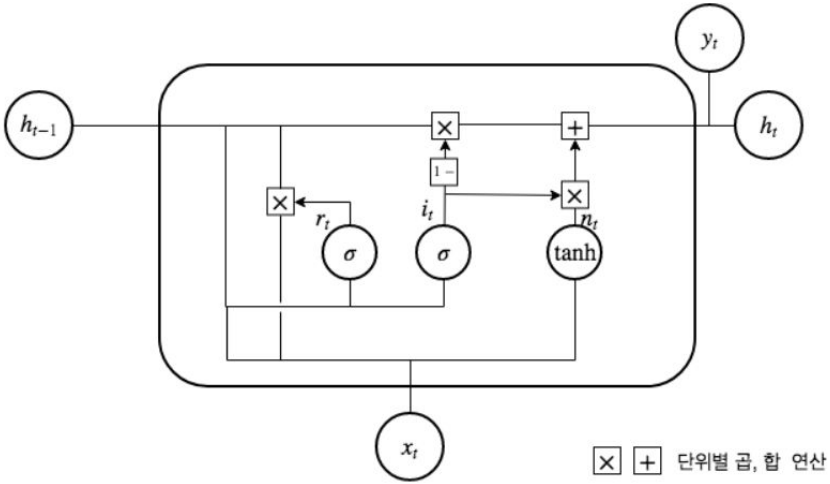
$$y_t = O(h_t)$$
$$h_t = R(h_{t-1}, x_t)$$

RNN Layer

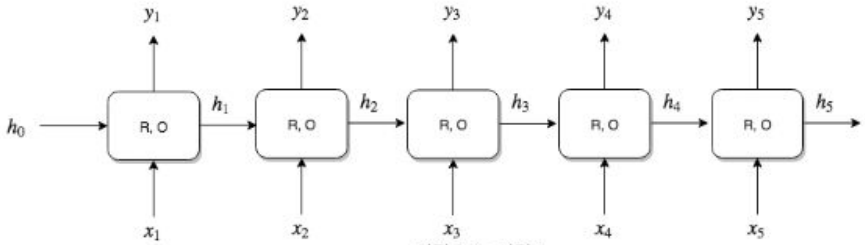


LSTM과 GRU

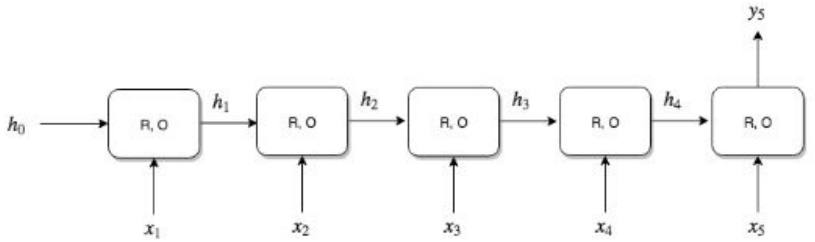

c_{t-1}
h_{t-1}
x_t
f_t
g_t
i_t
o_t
σ
tanh
c_t
h_t


h_{t-1}
x_t
y_t
h_t
r_t
i_t
n_t
σ
tanh
1 −
단위별 곱, 합 연산

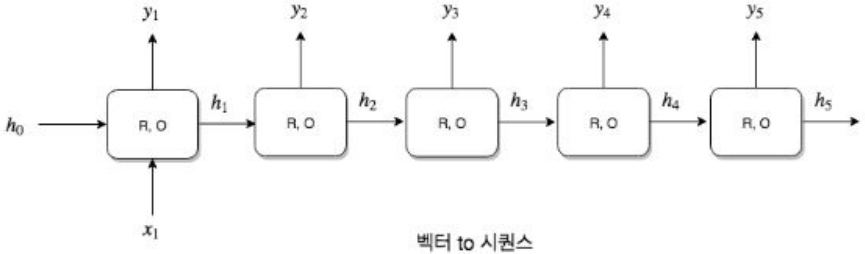
다양한 사용법



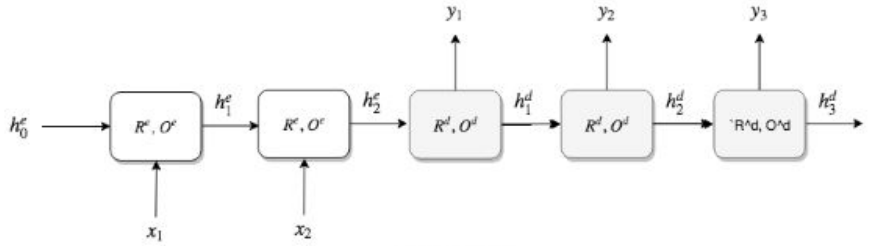
시퀀스 to 시퀀스



시퀀스 to 벡터



벡터 to 시퀀스



인코더 디코더

실습 : Entity Tagging and Intent Classification

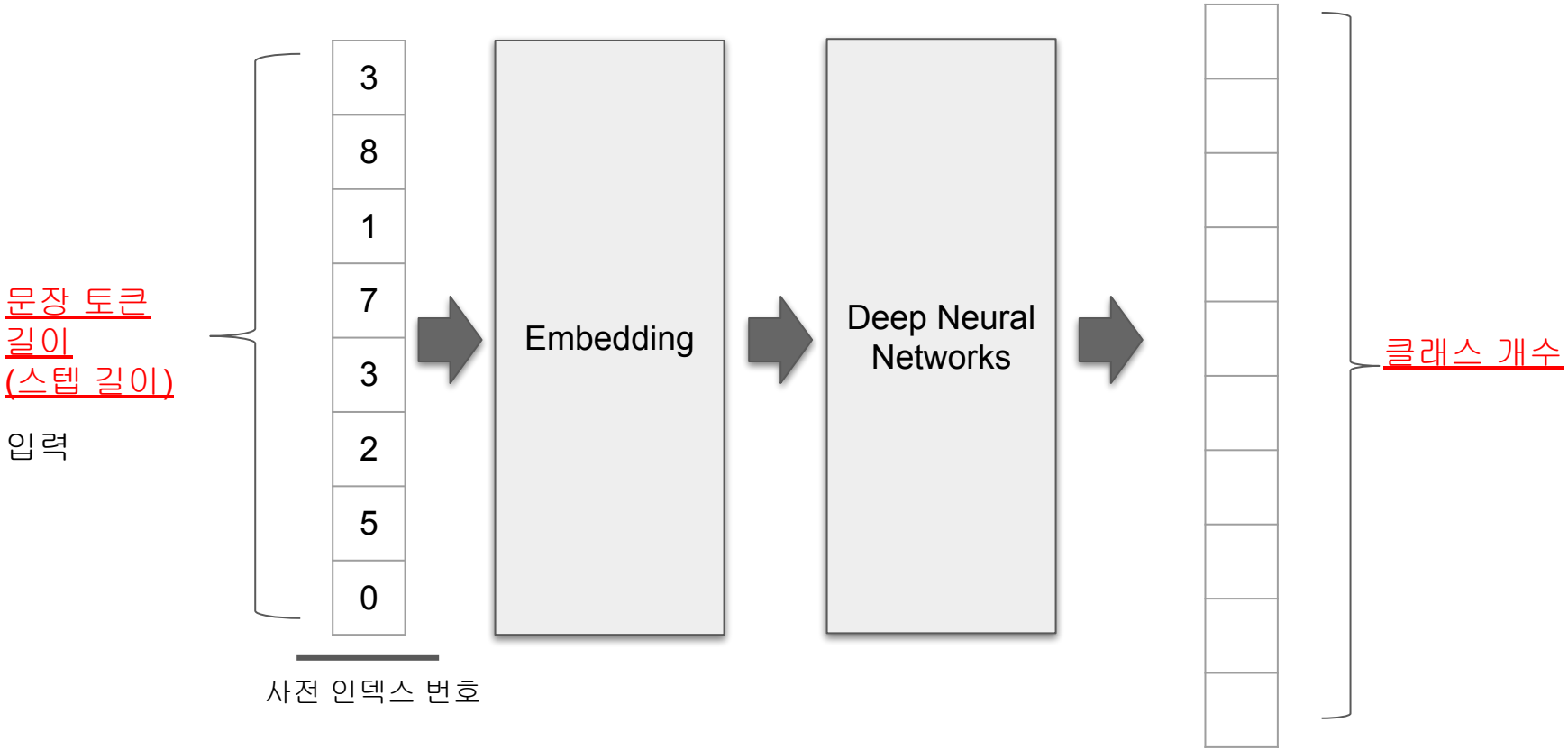
Height 송파 헬리오시티 구조물 높이 위키 피디아에서 뭐야
Facility

Intent 클래스 설계(10)

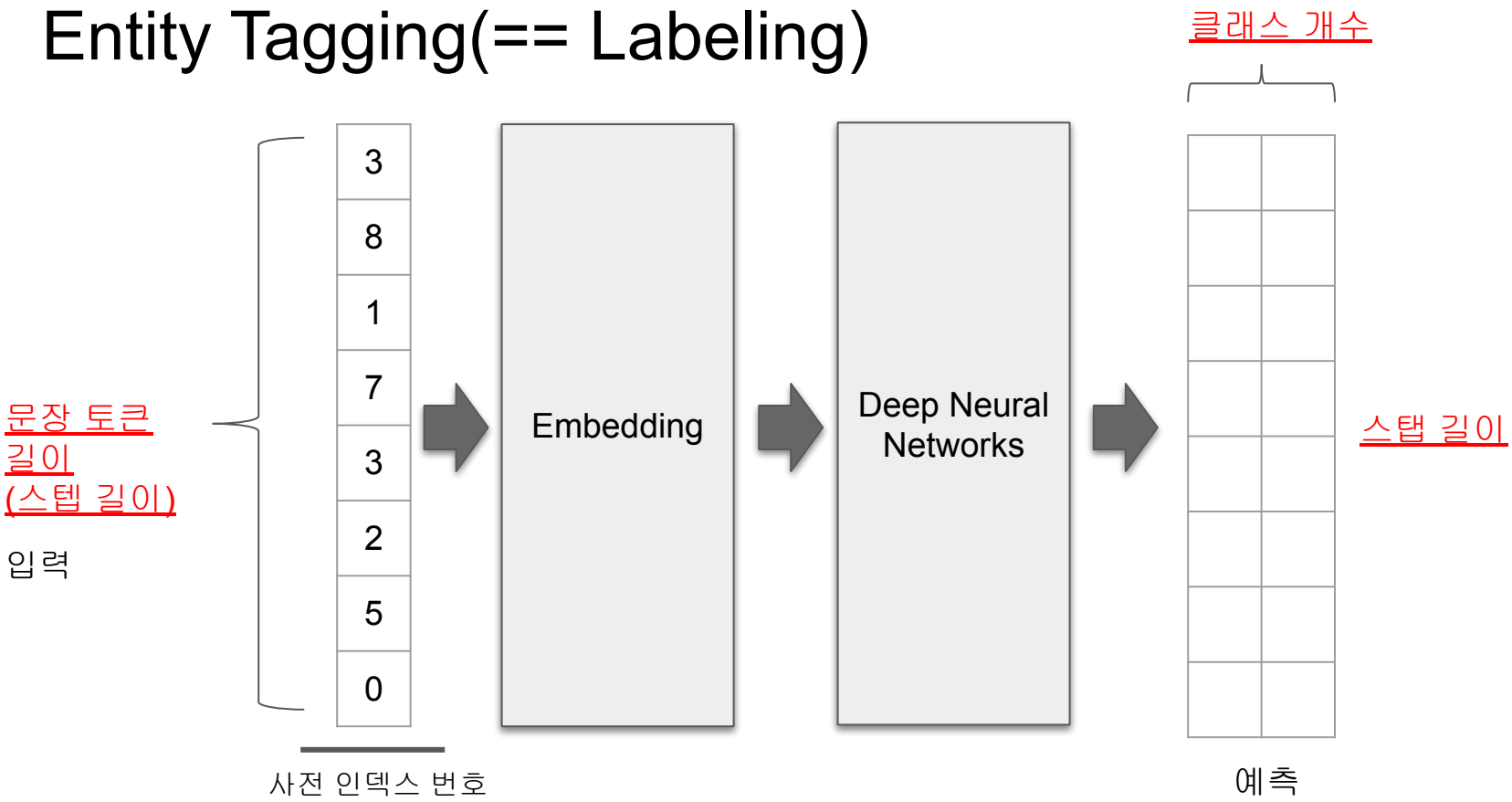
```
intent
age             900
area            900
belong_to       900
birth_date      900
birth_place     900
definition      900
height          900
length          900
weight          900
width           900
dtype: int64
```

- 실습 노트북
  - 3_1_intent_classification_pycon2019.ipynb
  - 3_2_entity_tagging_pycon2019.ipynb

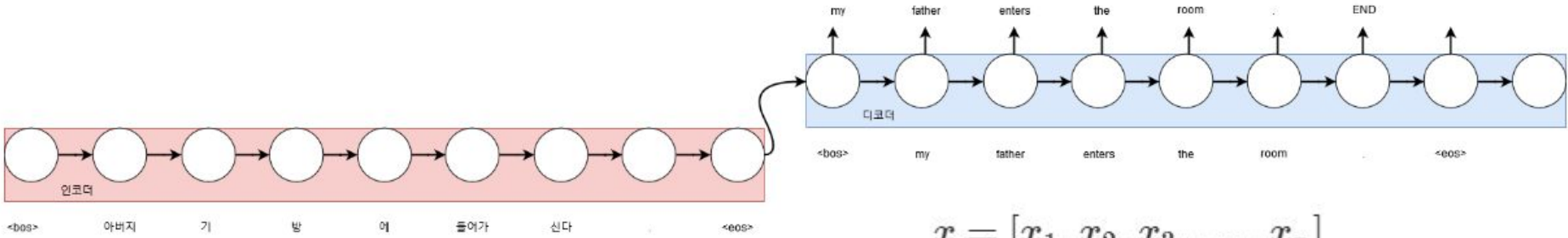
Intent Classification



Entity Tagging(== Labeling)



Attention



$$x = [x_1, x_2, x_3, \ldots, x_n]$$

$$y = [y_1, y_2, y_3, \ldots, y_m]$$

$$h_i = [\overrightarrow{h_i^T}; \overleftarrow{h_i^T}], i = 1, \ldots, n$$

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} \boldsymbol{h}_i$$; 출력 y_t에 대한 컨텍스트 벡터

$$\alpha_{t,i} = \text{align}(y_t, x_i)$$; 얼마나 두 단어 y_t 와 x_i 가 잘 매칭하는지...

$$= \frac{\text{score}(s_{t-1}, h_i)}{\sum_{i=1}^{n} \text{score}(s_{t-1}, h_i))}$$; 정렬 스코어를 소프트맥스로 계산.

Attention in GluonNLP

$$\text{score}(s_{t-1}, h_i) = s_{t-1}^\top h_i$$

$$\text{score}(s_{t-1}, h_i) = s_{t-1}^\top \mathbf{W}_a h_i$$

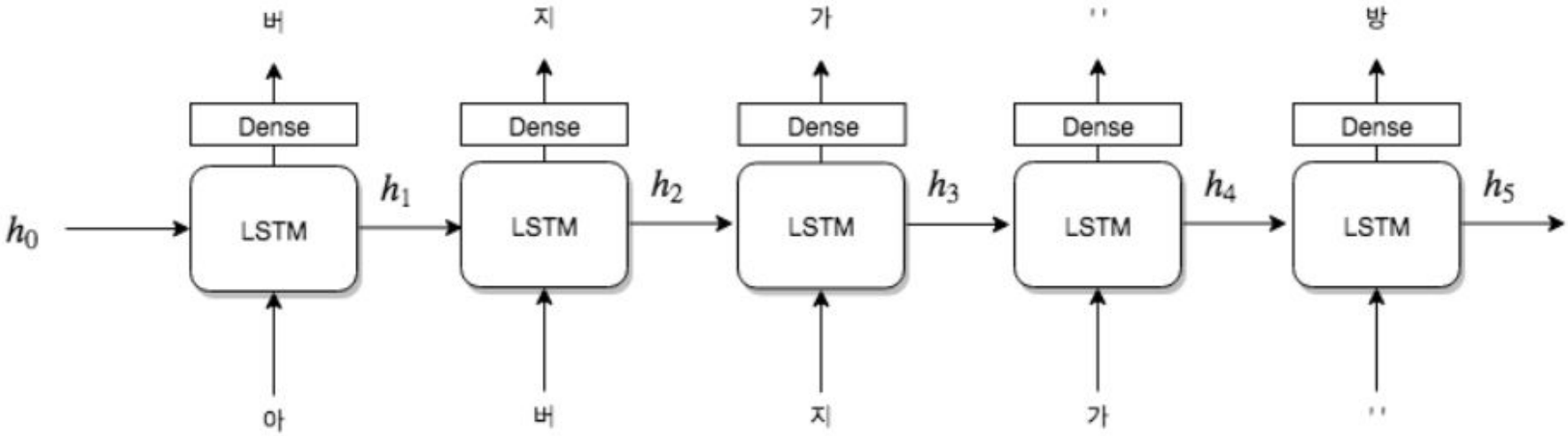
class **gluonnlp.model. DotProductAttentionCell**

class **gluonnlp.model. MLPAttentionCell**

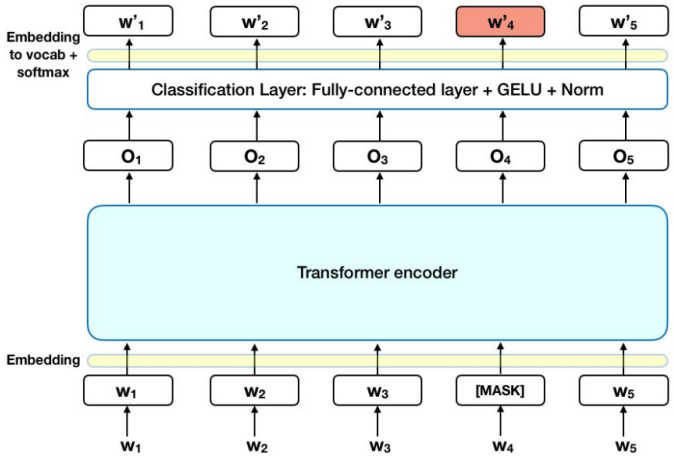
```
class EntityTagger(gluon.HybridBlock):
    def __init__(self, vocab_size, vocab_out_size, num_embed, hidden_size, use_attention=False, **kwargs):
        super(EntityTagger, self).__init__(**kwargs)
        self.hidden_size = hidden_size
        self.vocab_out_size = vocab_out_size
        self.use_attention = use_attention
        with self.name_scope():
            self.embed = nn.Embedding(input_dim=vocab_size, output_dim=num_embed)
            self.bigru = rnn.GRU(self.hidden_size, dropout=0.2, bidirectional=True)
            self.dense_prev = nn.Dense(10, flatten=False)
            self.dense = nn.Dense(self.vocab_out_size, flatten=False)
            if self.use_attention:
                self.attention = nlp.model.MLPAttentionCell(30, dropout=0.2)

    def hybrid_forward(self, F ,inputs, length):
        em_out = self.embed(inputs)
        bigruout = self.bigru(em_out)
        masked_encoded = F.SequenceMask(bigruout,
                                        sequence_length=length,
                                        use_sequence_length=True).transpose((1,0,2))
        if self.use_attention:
            masked_encoded,_ = self.attention(masked_encoded, masked_encoded)
        dense_out = self.dense_prev(masked_encoded)
        outs = self.dense(dense_out)
        return(outs)
```

Language Model

- P(아버지가 방에 들어가신다) >> P(아버지 가방에 들어가신다)
- Goal: 문장이나 단어들에 대한 확률을 구한다.
  - $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$
- 모델은 아래와 같은 두 형태를 주로 학습한다.
  - $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$
- 일종의 문법 모델, 그러나 언어모델(Language Model) LM이 표준



BERT(Bidirectional Embedding from Transformers)



- Multihead attention을 기반으로 한 Transformer 블록들의 네트워크
- 많은 양의 데이터로 전이학습(pre-training)
  - Masked LM, Next Sentence Prediction
- 적은 양의 데이터로 파인튜닝(fine-tuning)

from : [rani horev's blog : BERT explained])

양방향 RNN의 자기참조 이슈

- 양방향 RNN

- Masked LM

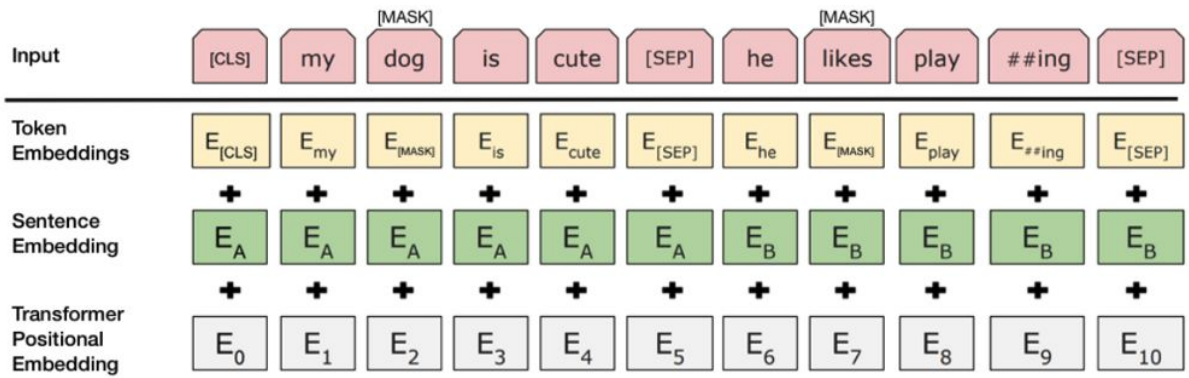
Next Sentence Prediction

- 문서에서 서로 인접한 문장들의 관계
  - 질문에 대한 답
  - 서술에 대한 부연설명
  - 원인과 결과
- 대량의 코퍼스를 기반으로 학습을 하게 됨으로써 다양한 문장의 다양한 관계를 학습하게 된다.

- 문장 A: 남자는 상점에 갔다.
- 문장 B: 팽귄은 날지 못한다.
- 레이블 : **다음문장 아님**

- 문장 A: 남자는 상점에 갔다.
- 문장 B: 1리터의 우유를 샀다.
- 레이블 : **다음문장임**

BERT Input



토큰 인덱스 : '[CLS] is this jack ##son ##ville ? [SEP] no it is not .[SEP]'

토큰 타입: 0 0 0 0 0 0 0 0 1 1 1 1 1 1

유효길이: 14

1. 각 토큰의 Vocabulary 인덱스를 추출해 이를 정해진 길이의 벡터로 생성.
2. 다수의 문장 구분을 위한 토큰 타입 벡터
3. 유효 길이 벡터

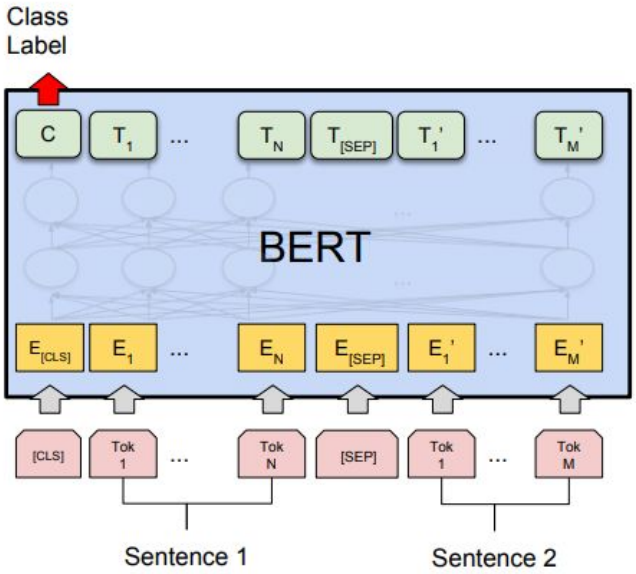
https://arxiv.org/pdf/1810.04805.pdf

BERT Input(2)

```
>>> bert_base, vocabulary = nlp.model.get_model('bert_12_768_12',
                                                dataset_name='wiki_multilingual_cased')
```

```
>>> ds = gluon.data.SimpleDataset([['나 보기가 역겨워', '김소월']])
>>> tok = nlp.data.BERTTokenizer(vocab=vocabulary, lower=False)
>>> trans = nlp.data.BERTSentenceTransform(tok, max_seq_length=10)

list(ds.transform(trans)
[(array([    2,  8982,  9356, 47869,  9566,     3,  8935, 22333, 38851, 3], dtype=int32),
  array(10, dtype=int32),
  array([0, 0, 0, 0, 0, 0, 1, 1, 1, 1], dtype=int32))]
```

Classification with BERT



(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG

https://arxiv.org/pdf/1810.04805.pdf

```
class BERTClassifier(nn.Block):
    def __init__(self,
                 bert,
                 num_classes=2,
                 dropout=None,
                 prefix=None,
                 params=None):
        super(BERTClassifier, self).__init__(prefix=prefix, params=params)
        self.bert = bert
        with self.name_scope():
            self.classifier = nn.HybridSequential(prefix=prefix)
            if dropout:
                self.classifier.add(nn.Dropout(rate=dropout))
            self.classifier.add(nn.Dense(units=num_classes))

    def forward(self, inputs, token_types, valid_length=None):
        _, class_label = self.bert(inputs, token_types, valid_length)
        return self.classifier(class_label)
```

BERT APIs in GluonNLP

- https://gluon-nlp.mxnet.io/api/modules/model.html#gluonnlp.model.bert_12_768_12
  - BioBERT, SciBERT,...
  - RoBERTa included in GluonNLP v.0.8

실습

- 3_3_naver_review_classifications_gluon_bert.ipynb