Deep Learning을 이용한 개체명 분석

정상근

2017-11-29

구현

개체명 분석

개체명 분석 문제 정의

개체명 분석

한국소비자보호원은 19일 시판중인 선물세트의 상당수가 과대 포장된 ….

ORGANIZATION **DATE**

Class 설계 (5개 Class)

- ✓ OG : Organization
- ✓ DT : Date
- ✓ TI : Time
- ✓ LC : Location
- ✓ PS : Person

개체명 분석 데이터 (1)

- https://github.com/krikit/annie

- 2016년 한글 및 한국어 정보처리 학술대회 : "2016년 엑소브레인 V2.0 & 울산대학교 말뭉치"라는 명칭의 CD로 배포

감성 분석 데이터 (2)

N21 Dataset

아	B-LC
프	I-LC
가	I-LC
니	I-LC
스	I-LC
탄	I-LC
의	O
	O
…	…
	O
헤	B-OG
즈	I-OG

Tab 문자로 구분 (\t)

- Train Data : ./applications/sentiment_analysis/data/ner.n2n.txt

BIO Tagging scheme for Sequential Data

B-*Tag*	Beginning of the *tag*
I-*Tag*	Inside of the *tag*
O	Outside

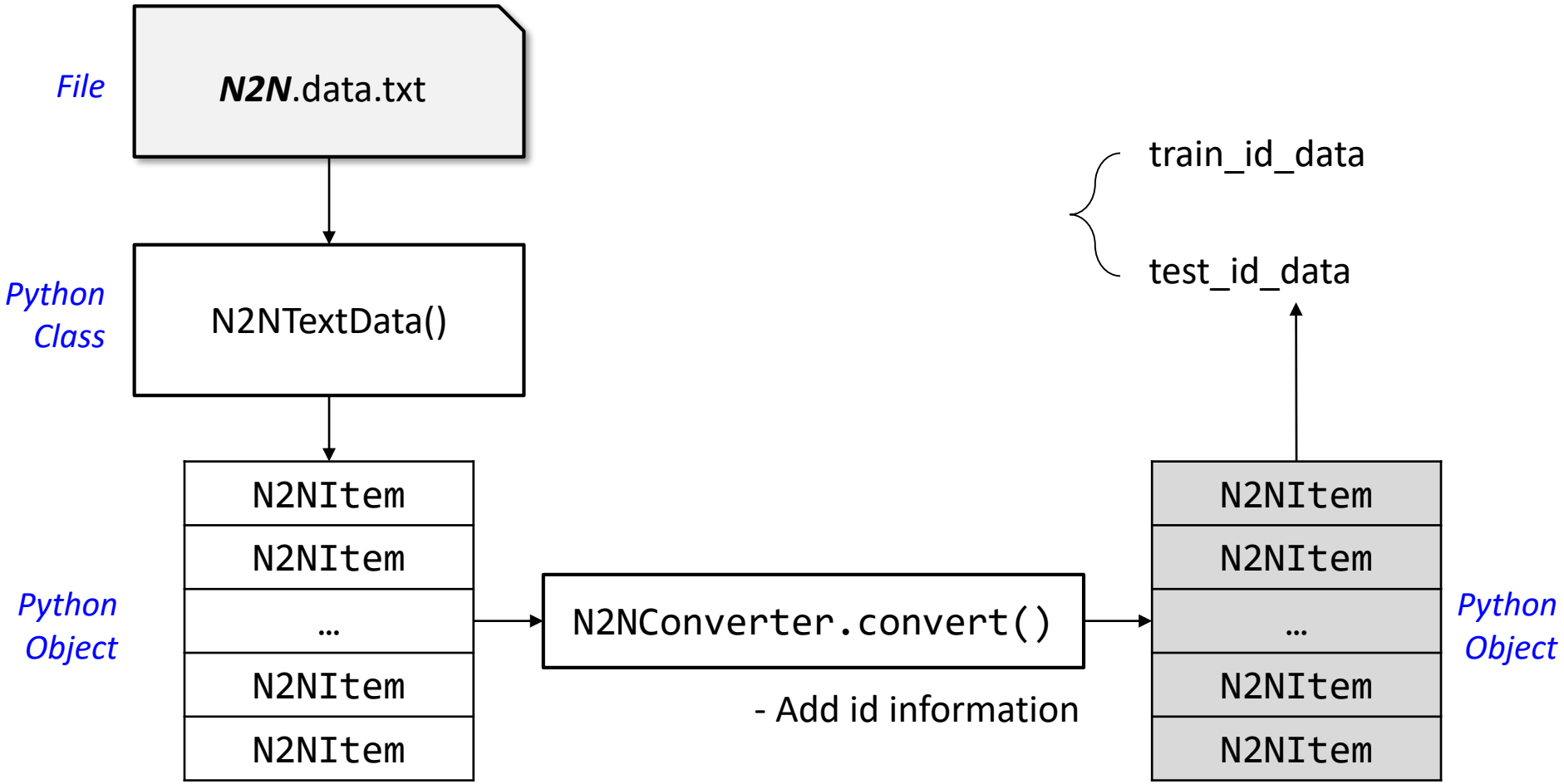
지	난	달		2	7	일	부	터		매	일		오	후		4	시	에
B-DT	**I-DT**	**I-DT**	**I-DT**	**I-DT**	**I-DT**	**I-DT**	**O**	**O**	**O**	**B-DT**	**I-DT**	**O**	**B-TI**	**I-TI**	**I-TI**	**I-TI**	**I-TI**	**O**

You can use other sequential tagging schems

- IO
- BMEWO
- BMEWO+
- BIOES
- …

https://lingpipe-blog.com/2009/10/14/coding-chunkers-as-taggers-io-bio-bmewo-and-bmewo/

[N2N] Data Processing Flow



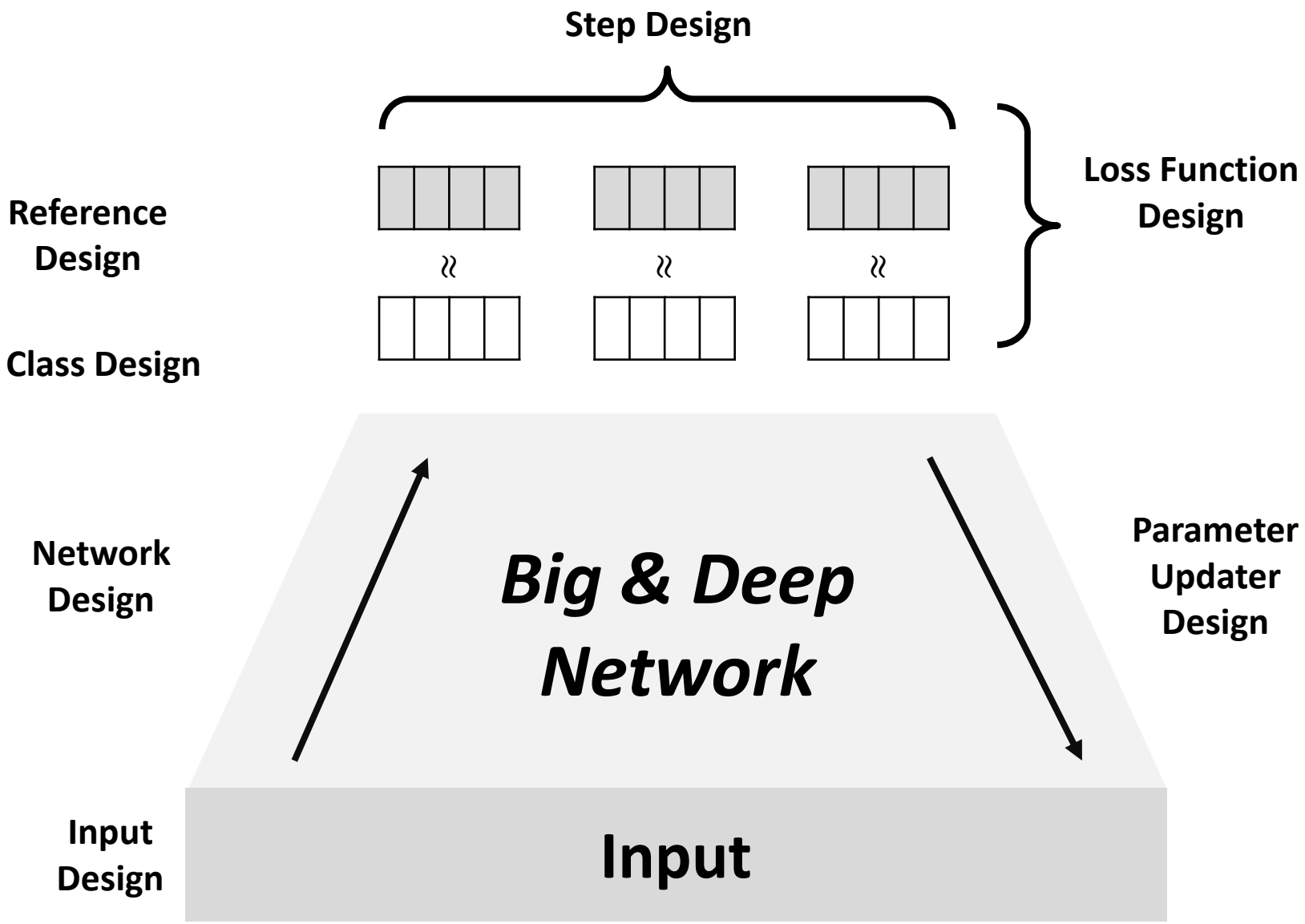
Code is available

https://github.com/hugman/deep_learning/tree/master/course/nlp

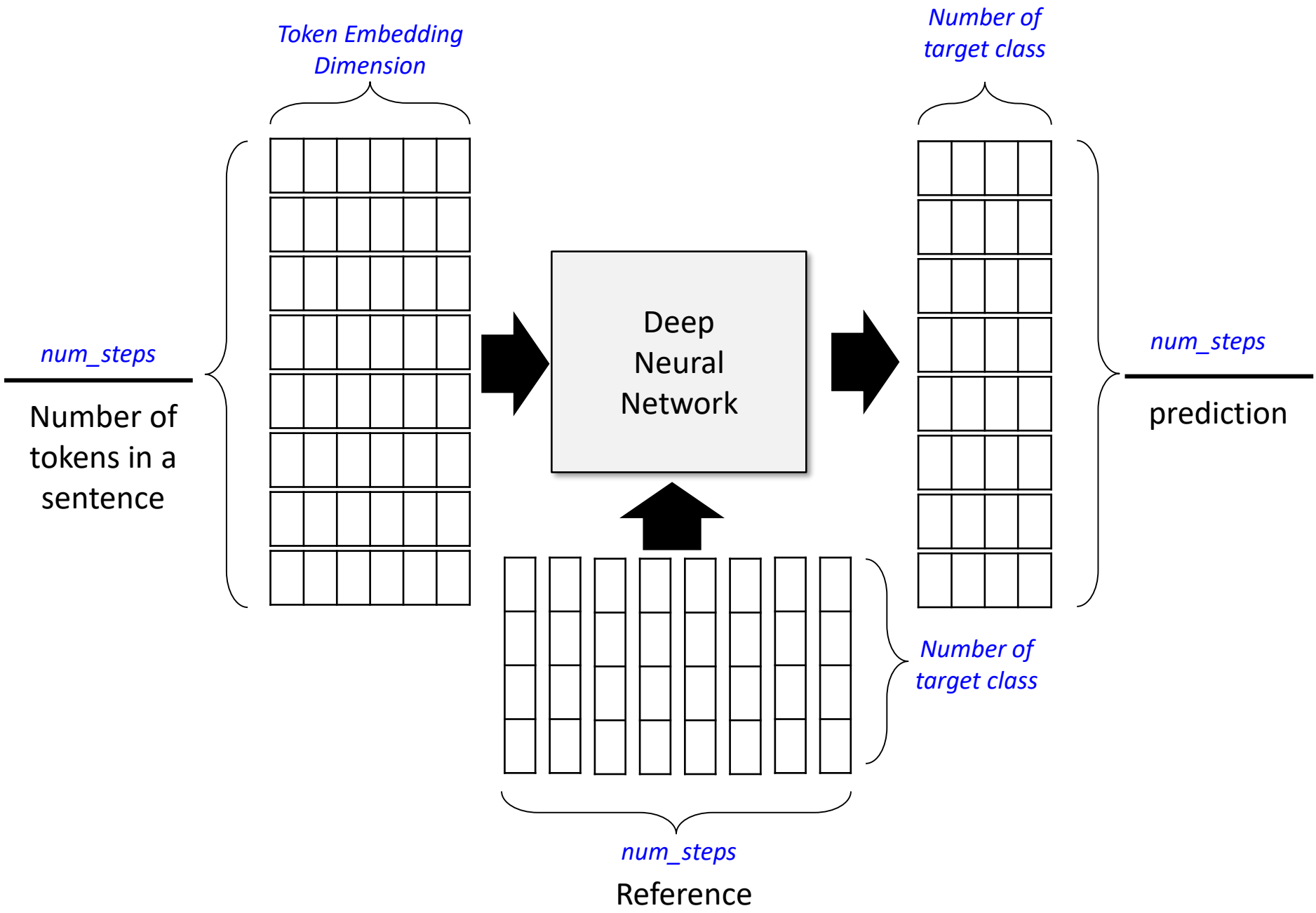
./applications./nanmed_entity_recognition/dataset : load_data()

```
# vocab loader
token_vocab_fn  = os.path.join( os.path.dirname(__file__), 'data',
'token.vocab.txt')
token_vocab     = Vocab(token_vocab_fn, mode='token')
target_vocab_fn = os.path.join( os.path.dirname(__file__), 'data',
'target.vocab.txt')
target_vocab    = Vocab(target_vocab_fn, mode='target')

# load train data
train_data_fn  = os.path.join( os.path.dirname(__file__), 'data', 'ner.n2n.txt')
train_txt_data = N2NTextData(train_data_fn)

# convert text data to id data
train_id_data  = N2NConverter.convert(train_txt_data, target_vocab, token_vocab)
```

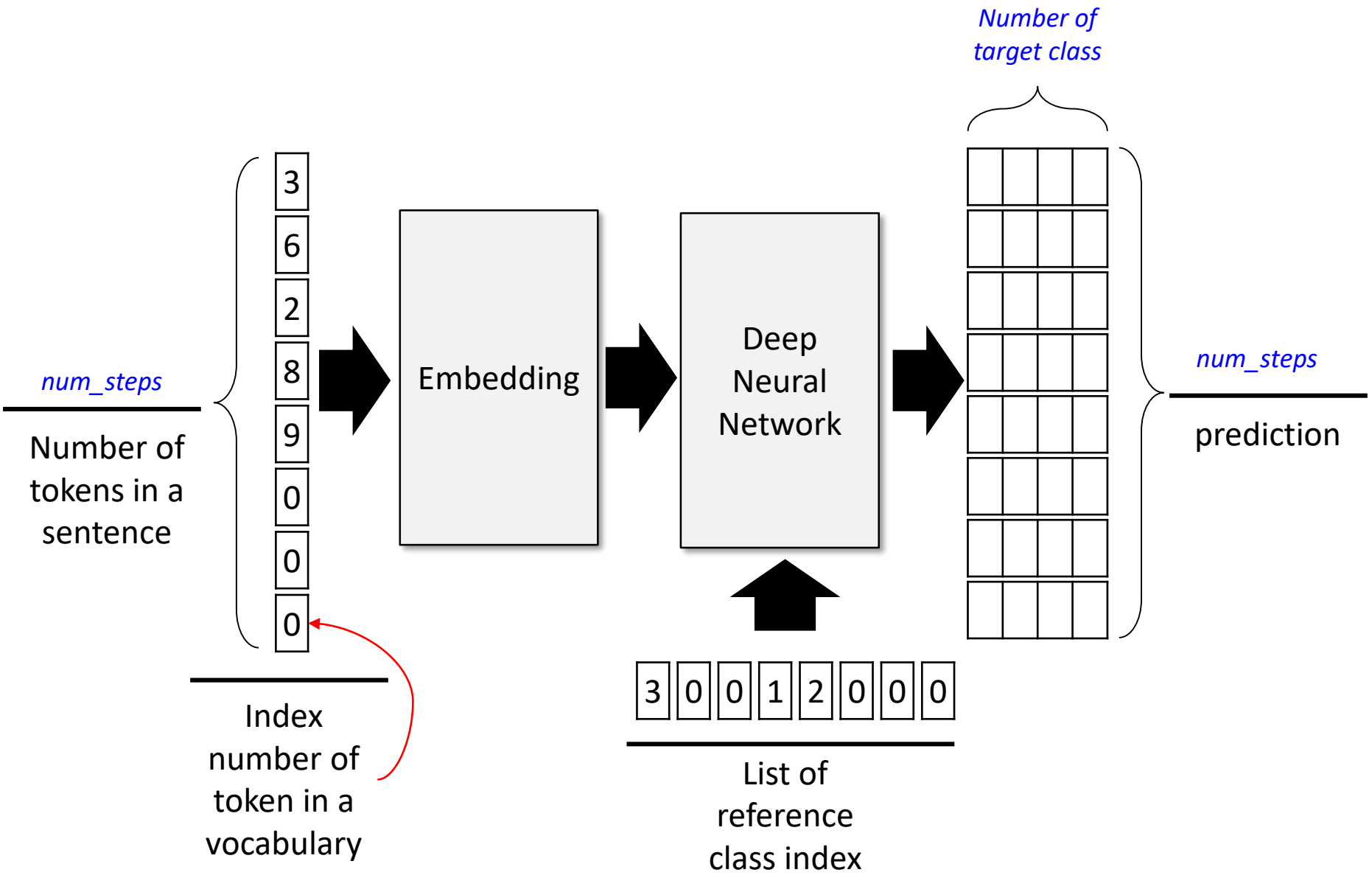
Building Block



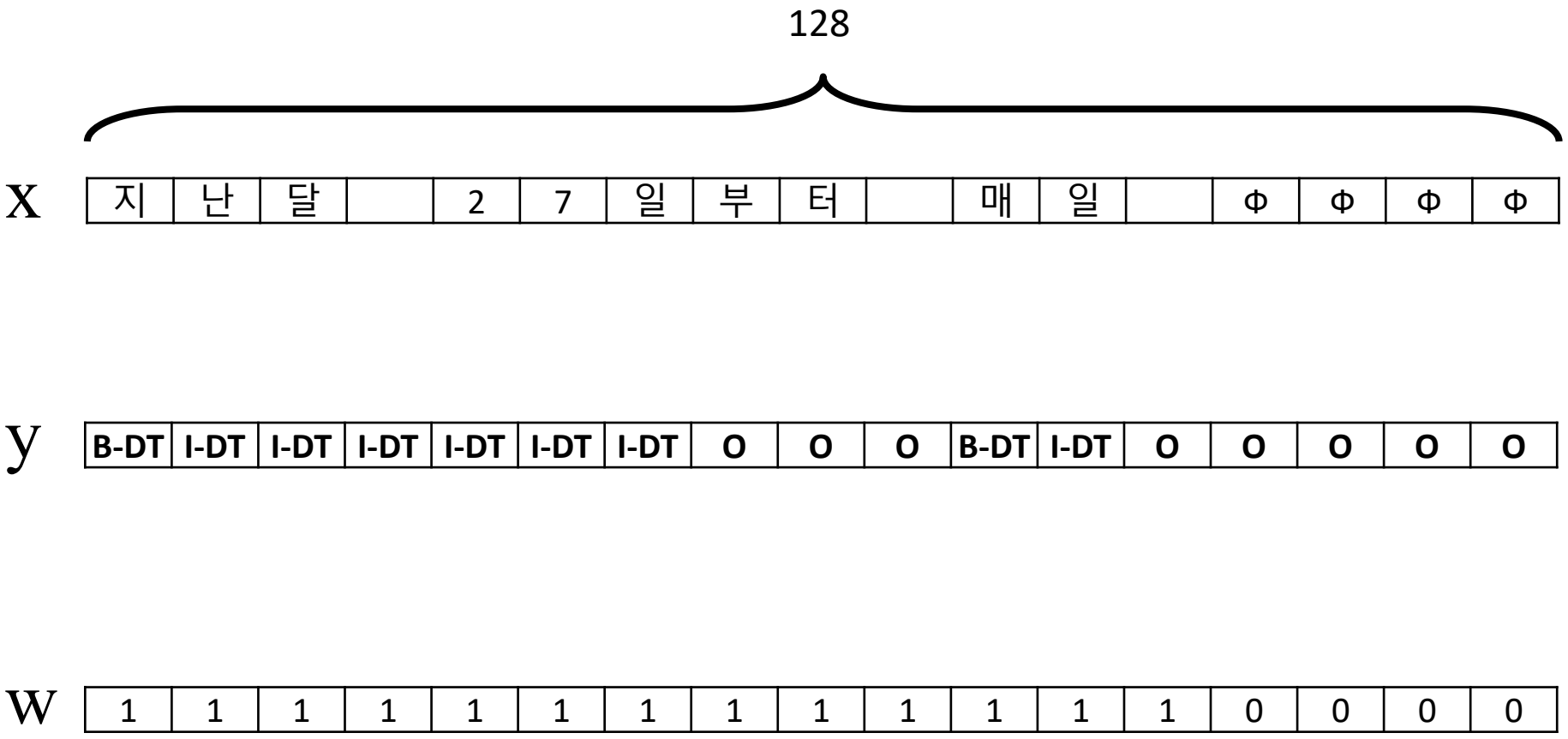
[N2N classification] DNN Interface



[N2N classification] DNN Interface with Embedding Library

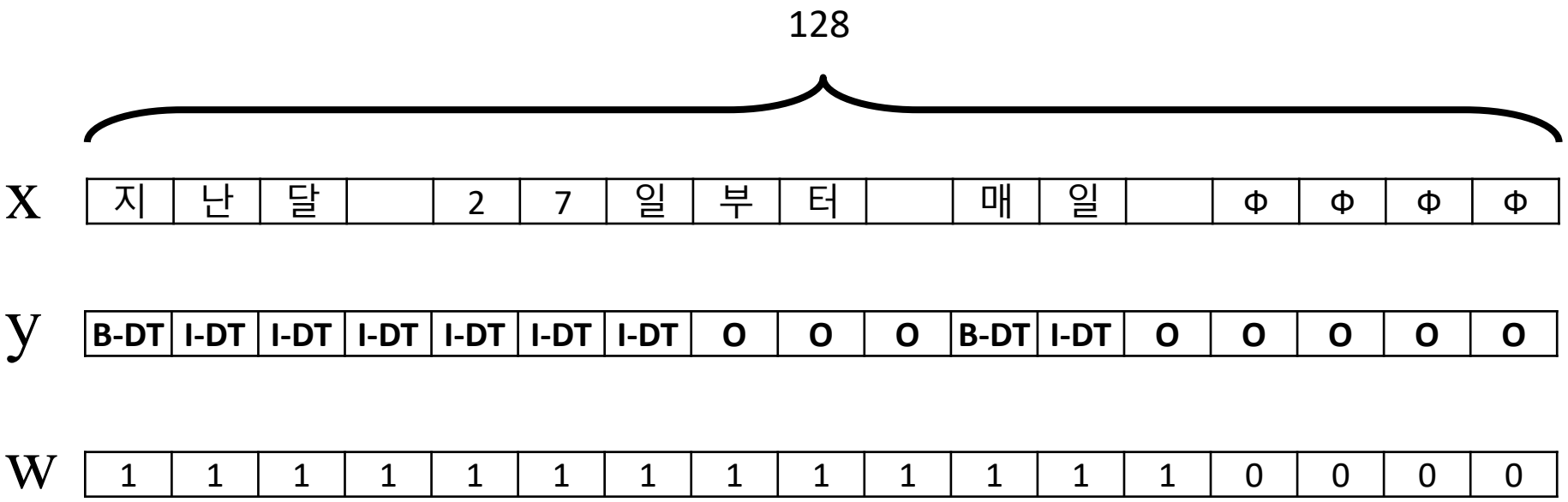


Input Design



Φ : padding symbol
W : to mark padding positions

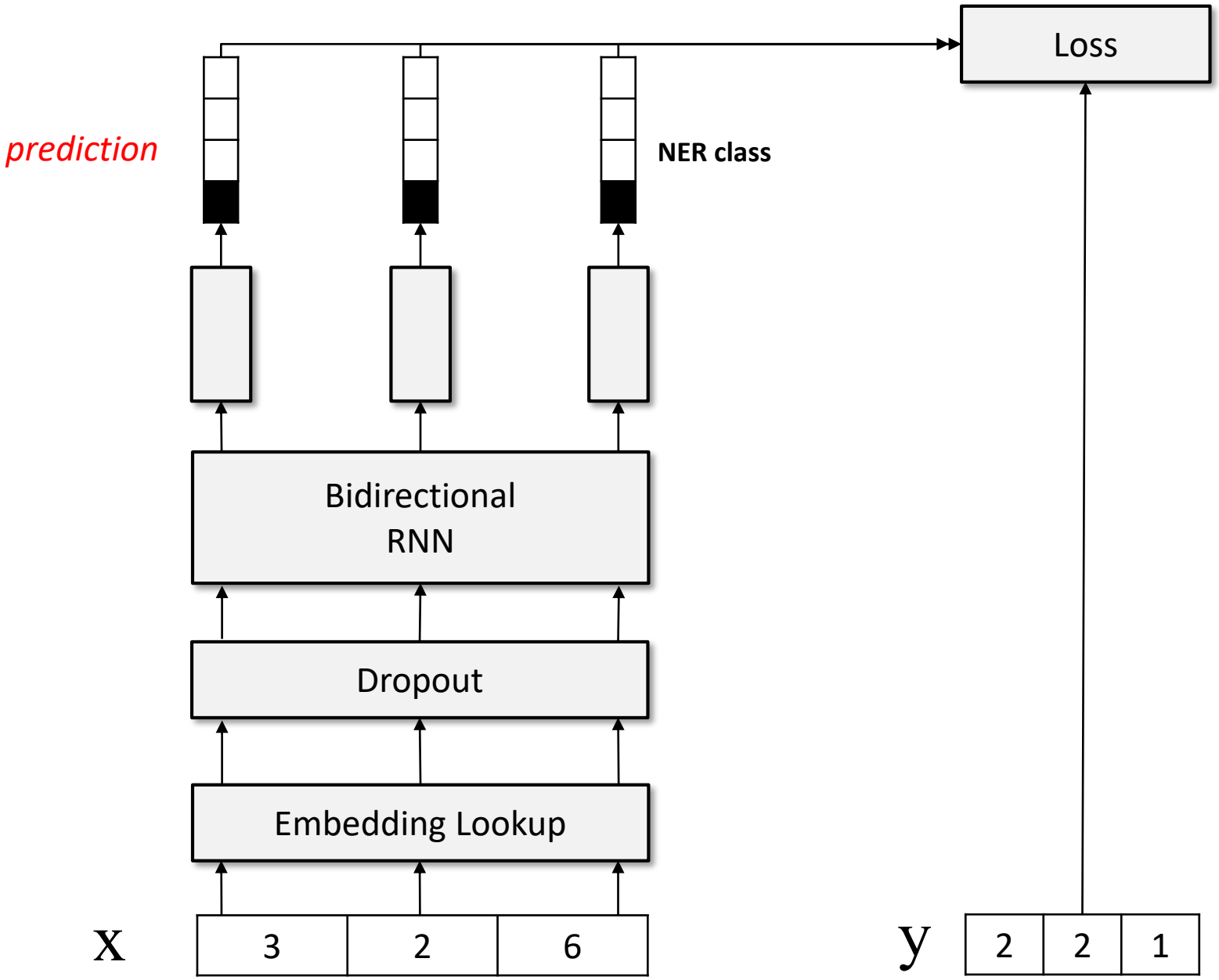
Tensorflow Interface Implementation



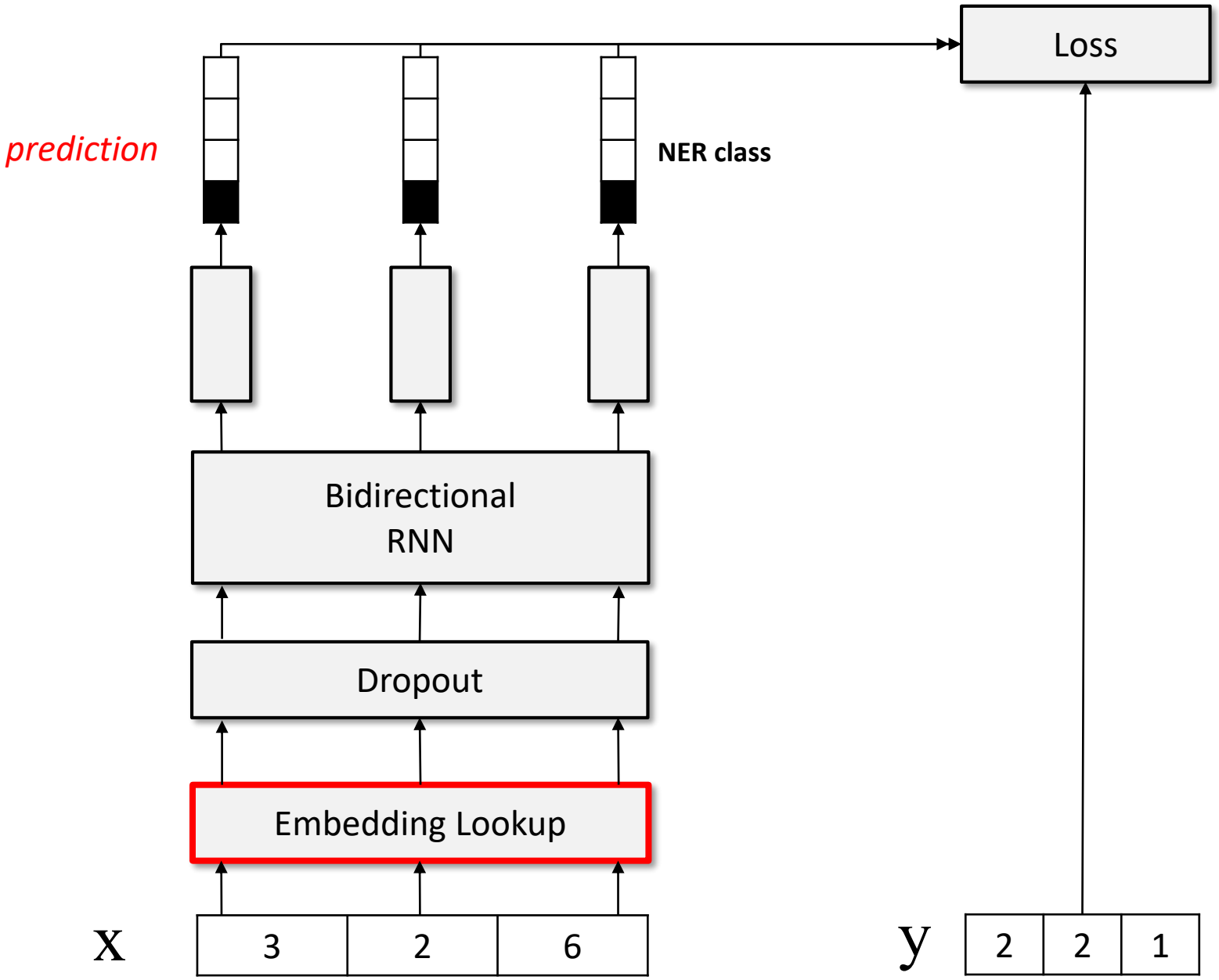

```
x = tf.placeholder(tf.int32, [None, hps.num_steps], name="pl_tokens")
y = tf.placeholder(tf.int32, [None, hps.num_steps], name="pl_target")
w = tf.placeholder(tf.int32, [None, hps.num_steps], name="pl_weight")
```

batch_size

Neural Network Design



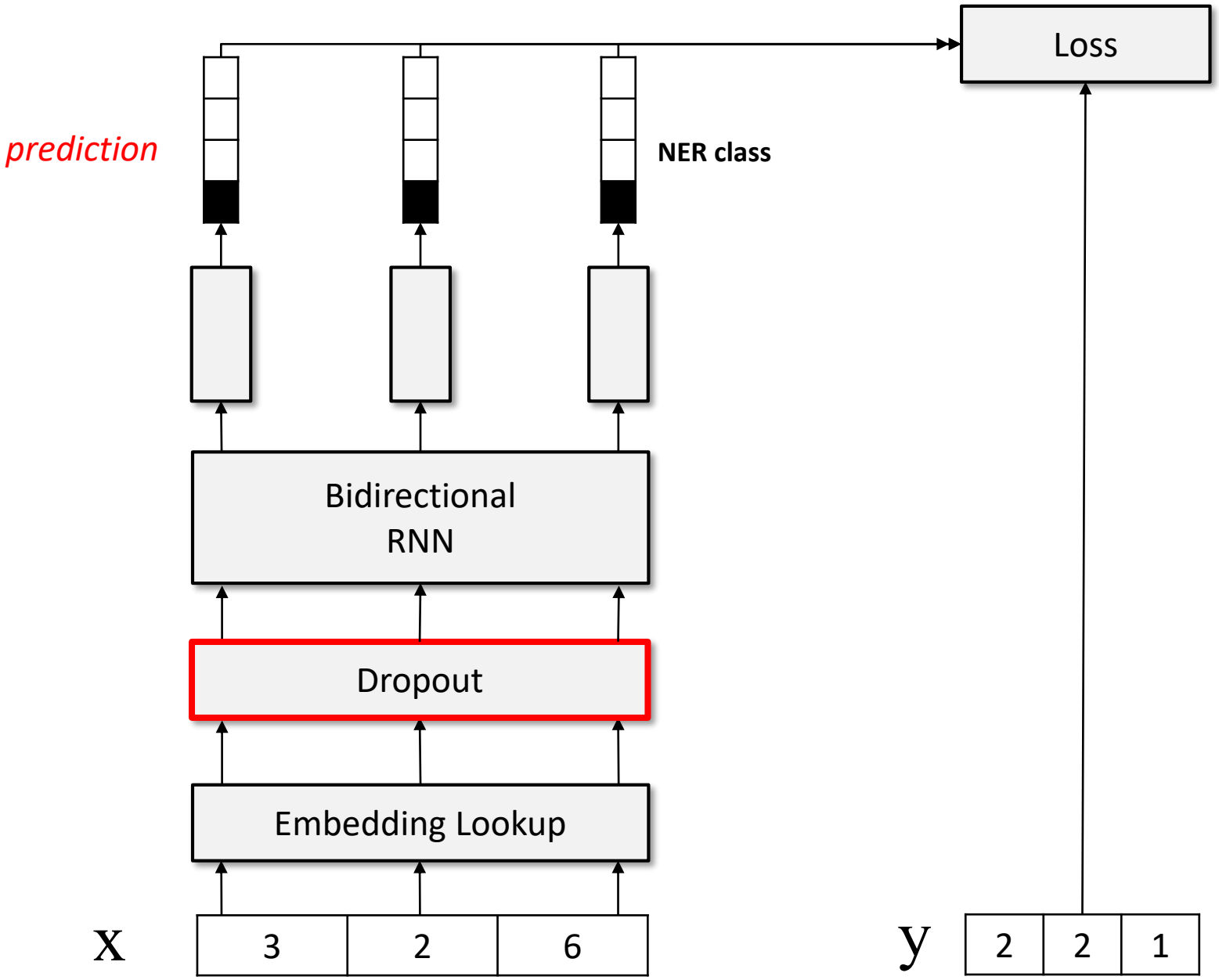
Neural Network Design | embedding



Tensorflow Implementation | Embedding

```
def _embedding(x)
    # character embedding
    shape       = [hps.vocab_size, hps.emb_size]
    initializer = tf.initializers.variance_scaling(distribution="uniform",
                                                   dtype=tf.float32)
    emb_mat     = tf.get_variable("emb", shape,
                                  initializer=initializer,
                                  dtype=tf.float32)

    # [batch_size, sent_len, emb_dim]
    input_emb   = tf.nn.embedding_lookup(emb_mat, x)

    # split input_emb -> num_steps
    step_inputs = tf.unstack(input_emb, axis=1)
```

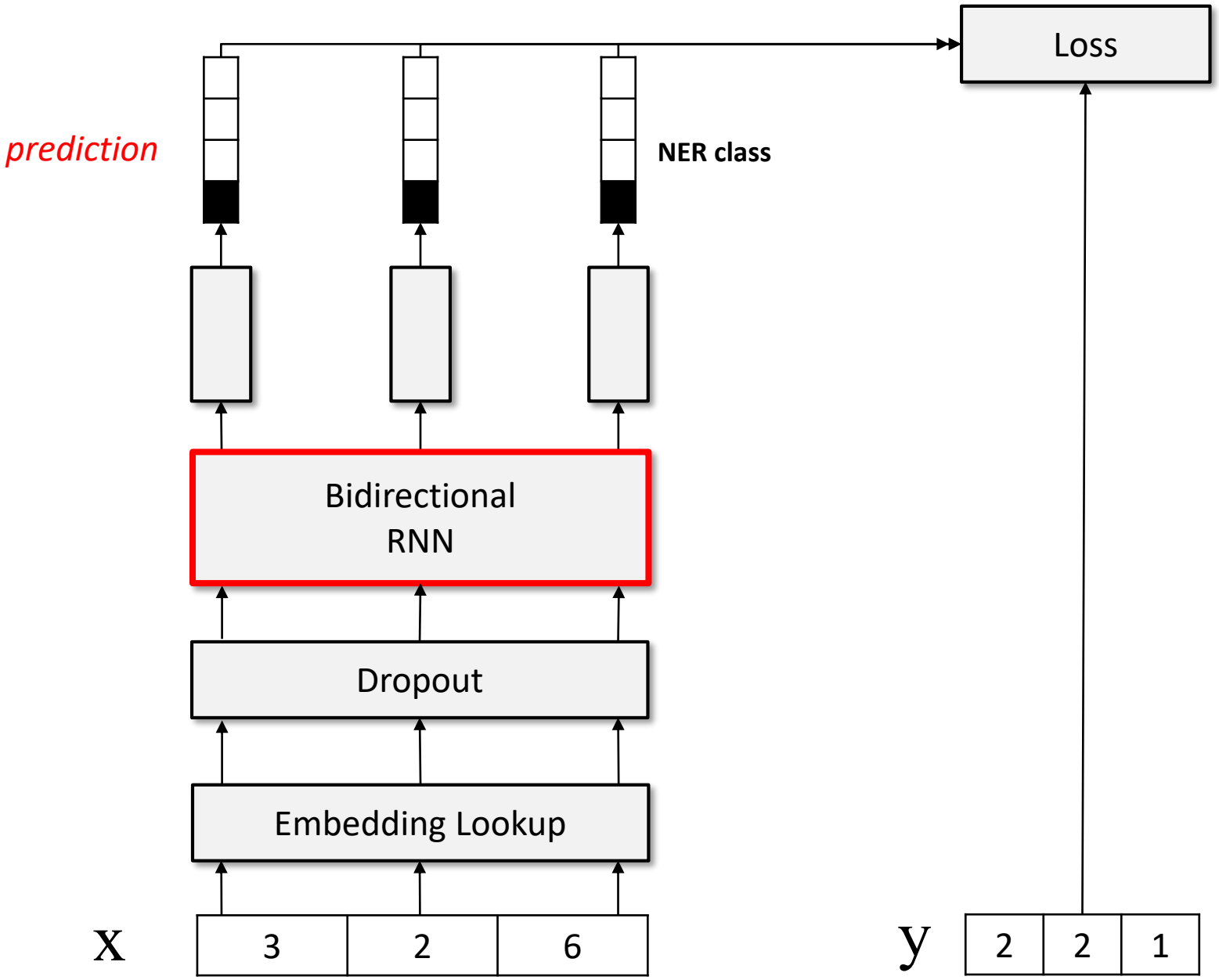
Neural Network Design



Tensorflow Implementation | Dropout

```
def _sequence_dropout(step_inputs, keep_prob)
    # apply dropout to each input
    # input : a list of input tensor which shape is [None, input_dim]
    with tf.name_scope('sequence_dropout') as scope:
            step_outputs = []
            for t, input in enumerate(step_inputs):
                    step_outputs.append( tf.nn.dropout(input, keep_prob) )
            return step_outputs
```

Neural Network Design | BiRNN based N2N encoding



Tensorflow Implementation | RNN based N2N encoding | Exercise

```
def sequence_encoding_n2n(step_inputs, seq_length, cell_size)
```

```
Input :  a list of <tf.Tensor shape=(?, 50), dtype=float32>
Return :  a list of <tf.Tensor, shape=(?, 100)>
```

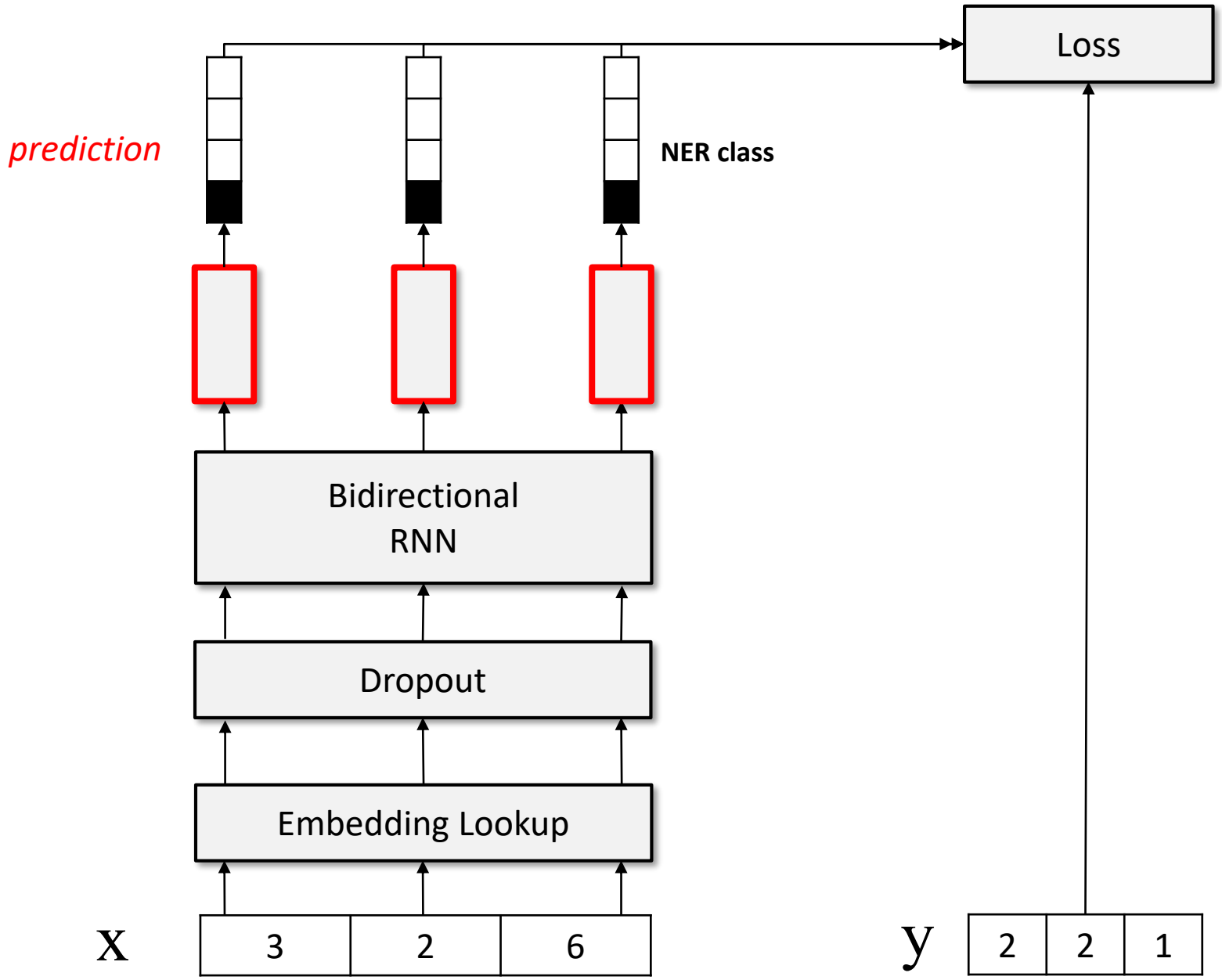
Keywords
tf.contrib.rnn.GRUCell
tf.nn.bidirectional_dynamic_rnn
tf.stack

Tensorflow Implementation | RNN based N2N encoding

```
def sequence_encoding_n2n(step_inputs, seq_length, cell_size)
```

```
# birnn based N2N encoding
f_rnn_cell = tf.contrib.rnn.GRUCell(cell_size, reuse=False)
b_rnn_cell = tf.contrib.rnn.GRUCell(cell_size, reuse=False)
_inputs    = tf.stack(step_inputs, axis=1)
outputs, states, = tf.nn.bidirectional_dynamic_rnn(
                                f_rnn_cell,
                                b_rnn_cell,
                                _inputs,
                                sequence_length=tf.cast(seq_length, tf.int64),
                                time_major=False,
                                dtype=tf.float32,
                                scope='birnn’,
                                )
output_fw, output_bw = outputs
states_fw, states_bw = states

output       = tf.concat([output_fw, output_bw], 2)
step_outputs = tf.unstack(output, axis=1)

final_state  = tf.concat([states_fw, states_bw], 1)
return step_outputs
```

Neural Network Design | to class N2N



Tensorflow Implementation | to class | Exercise

```
def _to_class(step_inputs, num_class)
```

```
Input :  a list of <tf.Tensor shape=(?, 200) dtype=float32>
Return :  a list of <tf.Tensor, shape=(?, 11), dtype=float32)>
```

Keywords
tensorflow.contrib.layers.python.layers.linear

Tensorflow Implementation | to class

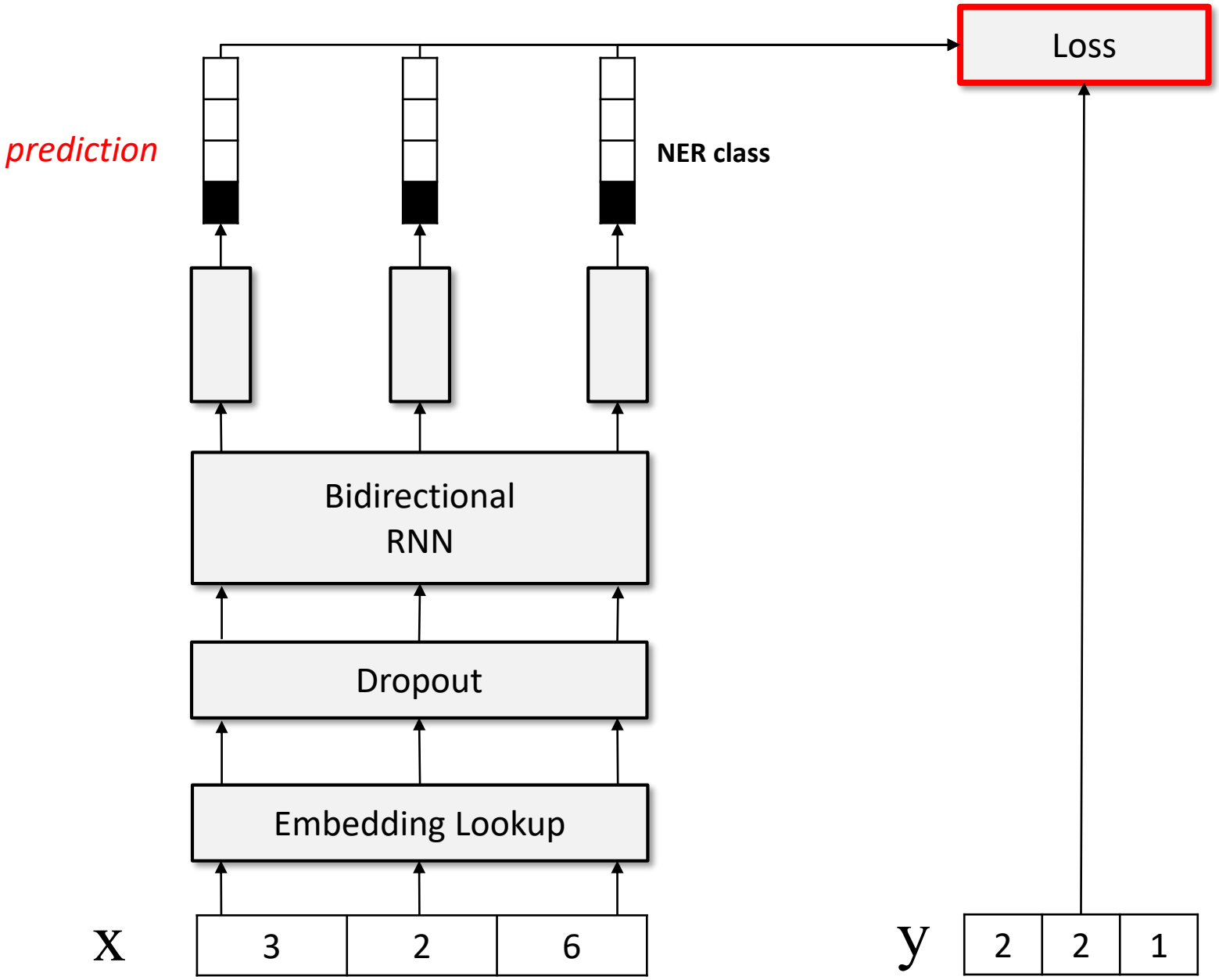
```
def _to_class(step_inputs, num_class)
```

```
T = len(step_inputs)
step_output_logits = []
for t in range(T):
        # encoder to linear(map)
        out = step_inputs[t]
        if t==0: out = linear(out, num_class, scope="Rnn2Target")
        else:    out = linear(out, num_class, scope="Rnn2Target", reuse=True)
        step_output_logits.append(out)
return step_output_logits
```

```
# output of the neural network
```

```
# step_preds and step_out_probs
step_out_probs = []
step_out_preds = []
for _output in step_outputs:
        _out_probs  = tf.nn.softmax(_output)
        _out_pred   = tf.argmax(_out_probs, 1)

        step_out_probs.append(_out_probs)
        step_out_preds.append(_out_pred)

# stack for interface
self.step_out_probs = tf.stack(step_out_probs, axis=1, name="step_out_probs")
self.step_out_preds = tf.stack(step_out_preds, axis=1, name="step_out_preds")
```

Neural Network Design | Loss calculation



Tensorflow Implementation | sequencewise loss calculation | Exercise

```
def _loss(step_outputs, step_refs, weights)
```

```
Input step_outputs:
   a list of <tf.tensor, shape=(?, 11), dtype=float32>
Input step_refs:
   a list of <tf.tensor, shape=(?, 128), dtype=int32>

Return :  tf.Tensor, shape=()
```

Keywords
tensorflow.contrib.seq2seq.sequence_loss

Tensorflow Implementation | sequencewise loss calculation

```
def _to_class_n2n(step_inputs, num_class)
    # step_outputs : a list of [batch_size, num_class] float32 - unscaled logits
    # step_refs    : [batch_size, num_steps] int32
    # weights      : [batch_size, num_steps] float32
    # calculate sequence wise loss function using cross-entropy
    _batch_output_logits = tf.stack(step_outputs, axis=1)
    loss = sequence_loss(
                            logits=_batch_output_logits,
                            targets=step_refs,
                            weights=weights
                        )
    return loss
```

- `from tensorflow.contrib.seq2seq import sequence_loss`

Tensorflow Implementation | parameter update

```
# optimizer settings

  optimizer       = tf.train.AdamOptimizer(hps.learning_rate)
  self.train_op   = optimizer.minimize(self.loss,
                                       global_step=self.global_step)
```

- ***train_op*** should be called outside of the network to update parameters

Training Process

```
# training process
while not sv.should_stop():
        fetches = [model.global_step, model.loss, model.train_op]
        a_batch_data = next( train_data_set.iterator )
        y, x, w = a_batch_data
        fetched = sess.run(fetches, {
                                                model.x: x,
                                                model.y: y,
                                                model.w: w,

                                                model.keep_prob: hps.keep_prob,
                                            }
                                )

        local_step += 1

        _global_step = fetched[0]
        _loss        = fetched[1]
```

Q/A

감사합니다.

Lecture Blog: www.hugman.re.kr
Lecture Video: https://goo.gl/7NL5hV
Lecture Slides: https://goo.gl/6NfR1V
Code Share: https://github.com/hugman
Facebook: https://goo.gl/1RML3C

정상근, Ph.D

Intelligence Architect

Senior Researcher, AI Tech. Lab. SKT Future R&D
Contact : hugmanskj@gmail.com, hugman@sk.com