NAVER NLP Challenge 2018

Semantic Role Labeling (SRL)

kozistr_team / HyeongChan Kim

kozistr@gmail.com

ME

kozistr

KOREATECH 재학

VOYAGERx Intern

http://kozistr.tech

뭐부터 하지??

Approach

1. Cell 부터 바꿔보자!

LSTMCell (Long Short-Term Memory)

GRUCell (Gated Recurrent Unit)

LayerNormBasicLSTMCell

CoupledInputForgetGateLSTMCell

GLSTMCell (Group Long Short-Term Memory)

NASCell (Neural Architecture Search)

SRUCell (Simple Recurrent Unit)

Approach

1. Cell 부터 바꿔보자!

LSTMCell (Long Short-Term Memory)

GRUCell (Gated Recurrent Unit)

LayerNormBasicLSTMCell

CoupledInputForgetGateLSTMCell → Best!

GLSTMCell (Group Long Short-Term Memory)

NASCell (Neural Architecture Search)

SRUCell (Simple Recurrent Unit)

Approach

2. 더 없나?!

tf.nn.rnn_cell.DropoutWrapper

Class **DropoutWrapper**

Inherits From: `RNNCell`

Aliases:

- Class `tf.contrib.rnn.DropoutWrapper`
- Class `tf.nn.rnn_cell.DropoutWrapper`

Defined in `tensorflow/python/ops/rnn_cell_impl.py`.

Operator adding dropout to inputs and outputs of the given cell.

tf.contrib.rnn.HighwayWrapper

☆☆☆☆☆

Class **HighwayWrapper**

Inherits From: `RNNCell`

Defined in `tensorflow/contrib/rnn/python/ops/rnn_cell.py`.

RNNCell wrapper that adds highway connection on cell input and output.

Based on: R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks", arXiv preprint arXiv:1505.00387, 2015. https://arxiv.org/abs/1505.00387

tf.nn.rnn_cell.ResidualWrapper

Class **ResidualWrapper**

Inherits From: `RNNCell`

Aliases:

- Class `tf.contrib.rnn.ResidualWrapper`
- Class `tf.nn.rnn_cell.ResidualWrapper`

Defined in `tensorflow/python/ops/rnn_cell_impl.py`.

RNNCell wrapper that ensures cell inputs are added to the outputs.

tf.contrib.rnn.AttentionCellWrapper

Class **AttentionCellWrapper**

Inherits From: `RNNCell`

Defined in `tensorflow/contrib/rnn/python/ops/rnn_cell.py`.

Basic attention cell wrapper.

Implementation based on https://arxiv.org/abs/1409.0473.

Approach

2. 더 없나?!

tf.nn.rnn_cell.DropoutWrapper

Class **DropoutWrapper**

Inherits From: `RNNCell`

Aliases:

- Class `tf.contrib.rnn.DropoutWrapper`
- Class `tf.nn.rnn_cell.DropoutWrapper`

Defined in `tensorflow/python/ops/rnn_cell_impl.py`.

Operator adding dropout to inputs and outputs of the given cell.

tf.contrib.rnn.HighwayWrapper

☆☆☆☆☆

Class **HighwayWrapper**

Inherits From: `RNNCell`

Defined in `tensorflow/contrib/rnn/python/ops/rnn_cell.py`.

RNNCell wrapper that adds highway connection on cell input and output.

Based on: R. K. Srivastava, K. Greff, and J. Schmidhuber, "Highway networks", arXiv preprint arXiv:1505.00387, 2015. https://arxiv.org/abs/1505.00387

tf.nn.rnn_cell.ResidualWrapper

Class **ResidualWrapper**

Inherits From: `RNNCell`

Aliases:

- Class `tf.contrib.rnn.ResidualWrapper`
- Class `tf.nn.rnn_cell.ResidualWrapper`

Defined in `tensorflow/python/ops/rnn_cell_impl.py`.

RNNCell wrapper that ensures cell inputs are added to the outputs.

tf.contrib.rnn.AttentionCellWrapper

Class **AttentionCellWrapper**

Inherits From: `RNNCell`

Defined in `tensorflow/contrib/rnn/python/ops/rnn_cell.py`.

Basic attention cell wrapper.

Implementation based on https://arxiv.org/abs/1409.0473.

Approach

2. 더 없나?!

tf.nn.rnn_cell.DropoutWrapper

Class **DropoutWrapper**

Inherits From: `RNNCell`

Aliases:

- Class `tf.contrib.rnn.DropoutWrapper`
- Class `tf.nn.rnn_cell.DropoutWrapper`

Defined in `tensorflow/python/ops/rnn_cell_impl.py`.

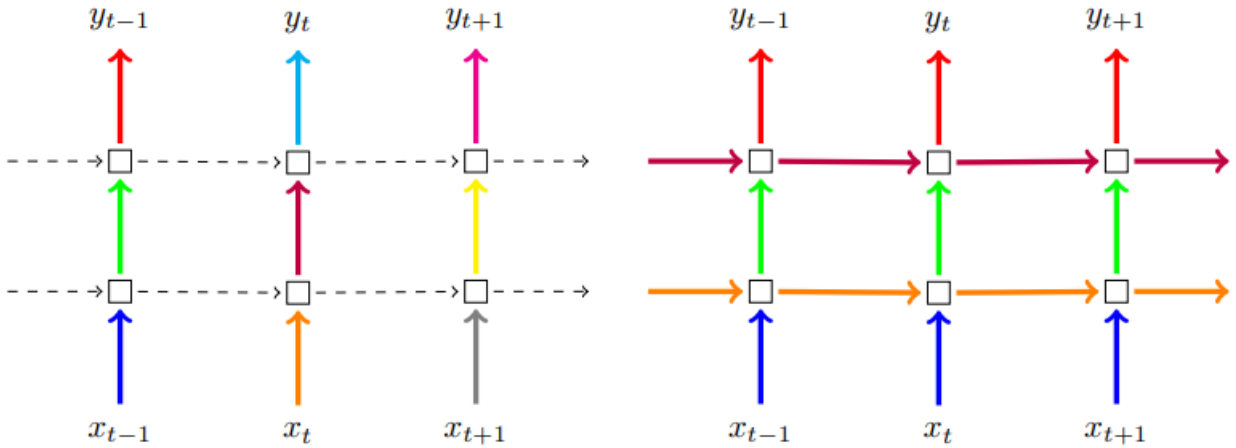
Operator adding dropout to inputs and outputs of the given cell.

https://arxiv.org/pdf/1512.05287.pdf

__init__

```
__init__(
    cell,
    input_keep_prob=1.0,
    output_keep_prob=1.0,
    state_keep_prob=1.0,
    variational_recurrent=False,
    input_size=None,
    dtype=None,
    seed=None,
    dropout_state_filter_visitor=None
)
```

Using variational_recurrent



(a) Naive dropout RNN (b) Variational RNN

Approach

3. 순서는?!

CIFGLSTMCell	CIFGLSTMCell	CIFGLSTMCell	CIFGLSTMCell
HighwayWrapper	HighwayWrapper	ResidualWrapper	ResidualWrapper
DropoutWrapper	ResidualWrapper	HighwayWrapper	DropoutWrapper
ResidualWrapper	DropoutWrapper	DropoutWrapper	HighwayWrapper

Approach

3. 순서는?!

CIFGLSTMCell
HighwayWrapper
DropoutWrapper
ResidualWrapper

Best!

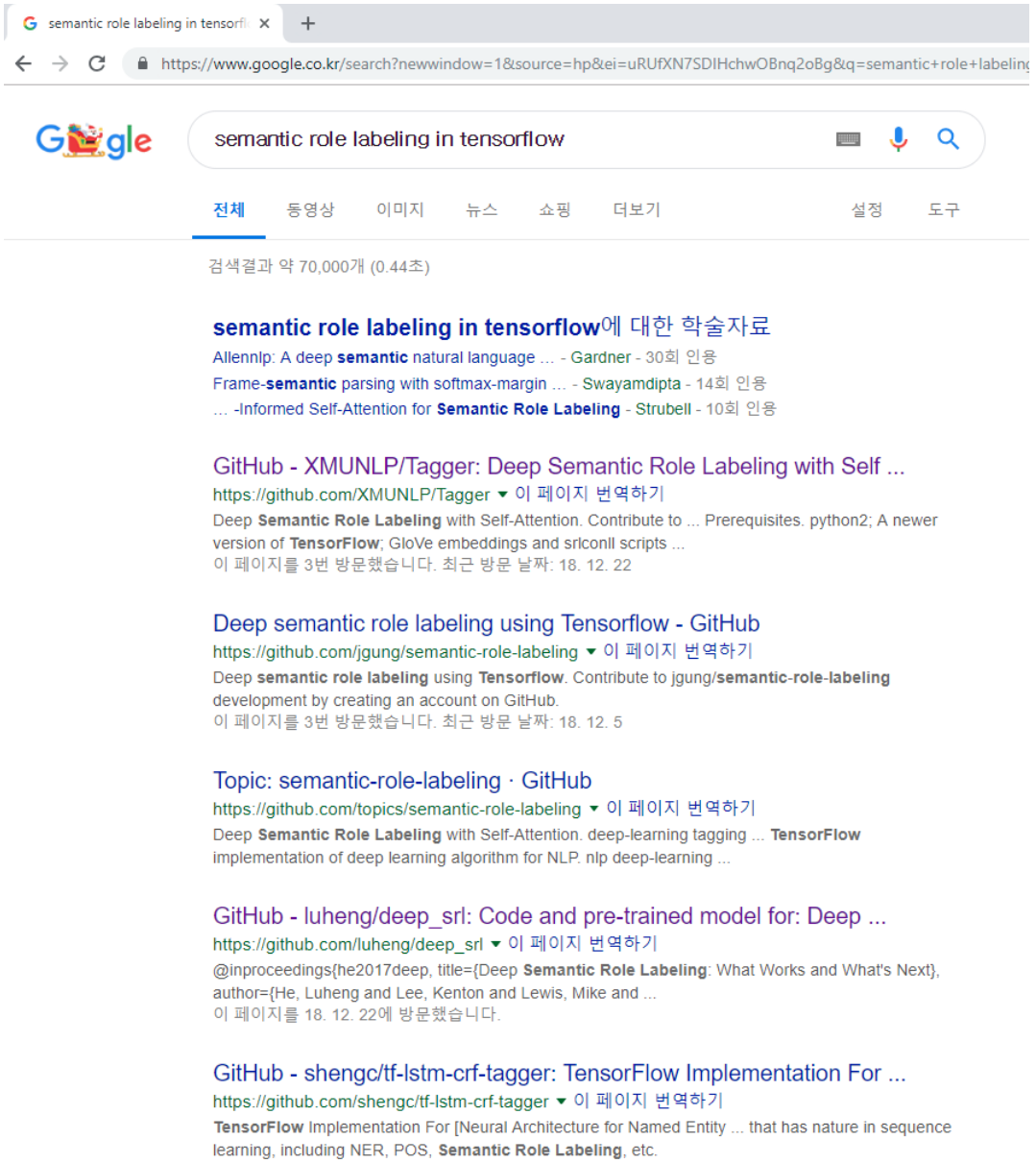
CIFGLSTMCell
HighwayWrapper
ResidualWrapper
DropoutWrapper

CIFGLSTMCell
ResidualWrapper
HighwayWrapper
DropoutWrapper

CIFGLSTMCell
ResidualWrapper
DropoutWrapper
HighwayWrapper

Approach

4. 이제 모델 건드려 보자!



Approach

4. 이제 모델 건드려 보자!

1 layer bi-LSTM (baseline)	2 layers stacked LSTM w/ reversed sequence
self attention	multi-head attention
Transformer	BERT?!
Densely-Connected bi-LSTM	highway + Densely-Connected bi-LSTM

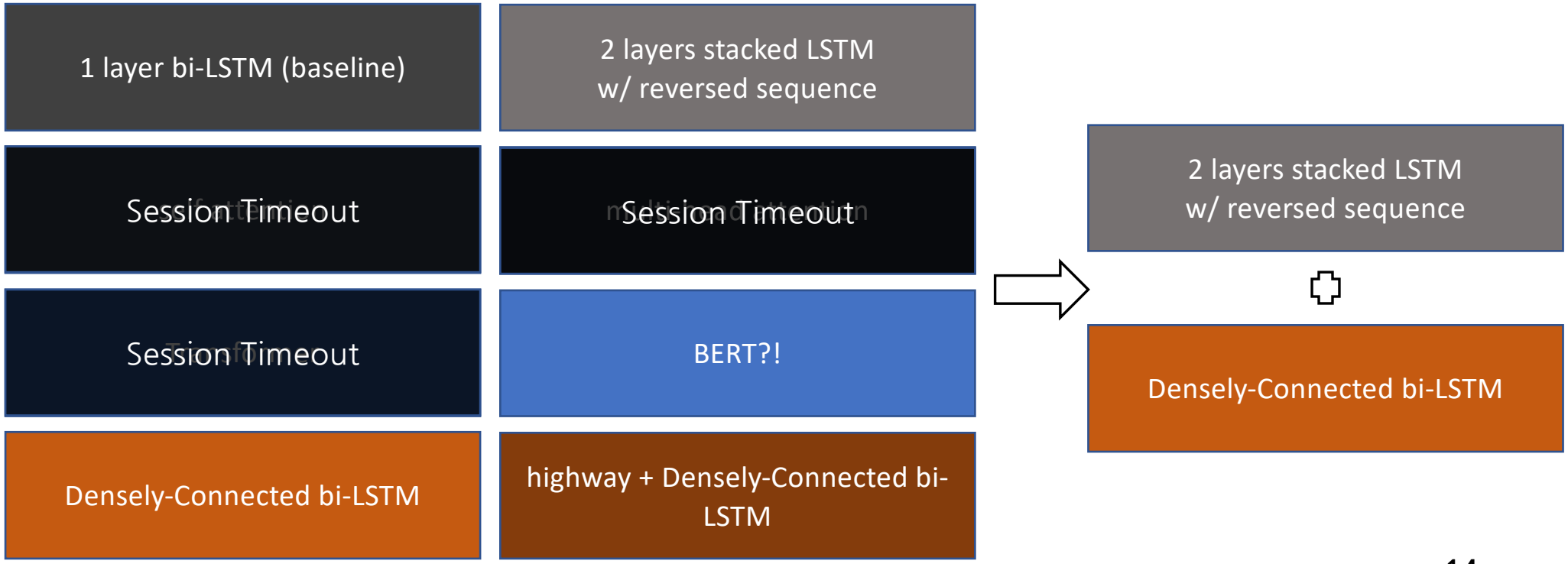
Approach

4. 이제 모델 건드려 보자!

1 layer bi-LSTM (baseline)	2 layers stacked LSTM w/ reversed sequence
Session Timeout	Session Timeout
Session Timeout	BERT?!
Densely-Connected bi-LSTM	highway + Densely-Connected bi-LSTM

Approach

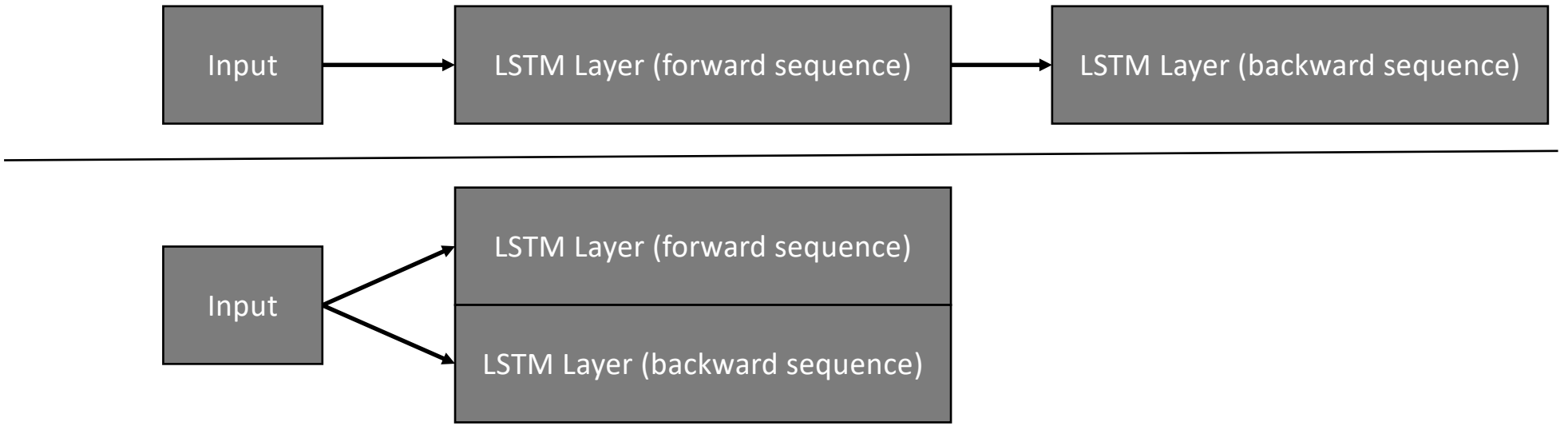
4. 이제 모델 건드려 보자!



Approach

4. 이제 모델 건드려 보자!

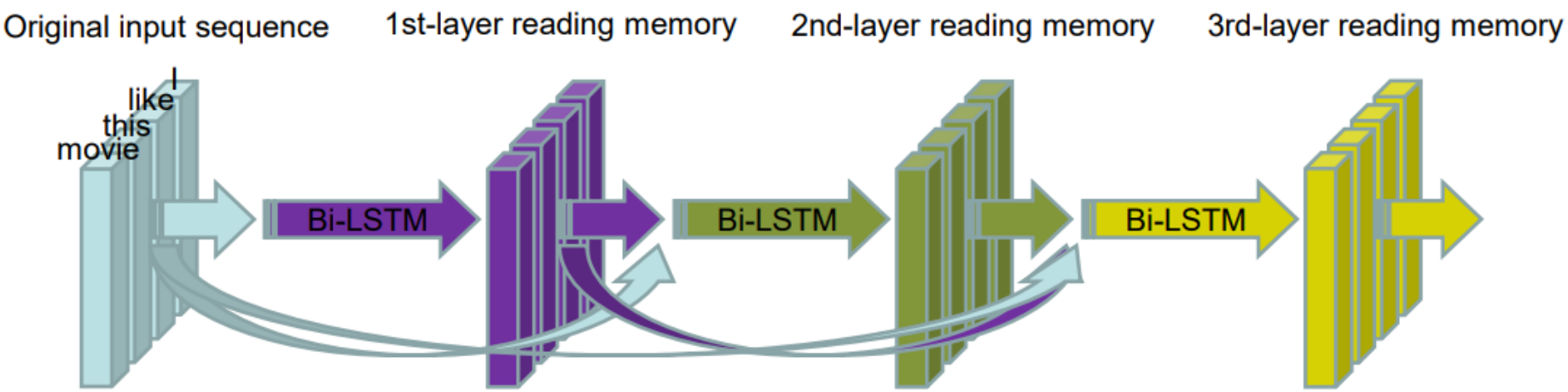
2 layers stacked LSTM w/ reversed sequence (bi-LSTM wisely)



Approach

4. 이제 모델 건드려 보자!

Densely-Connected bi-LSTM



https://arxiv.org/abs/1802.00889

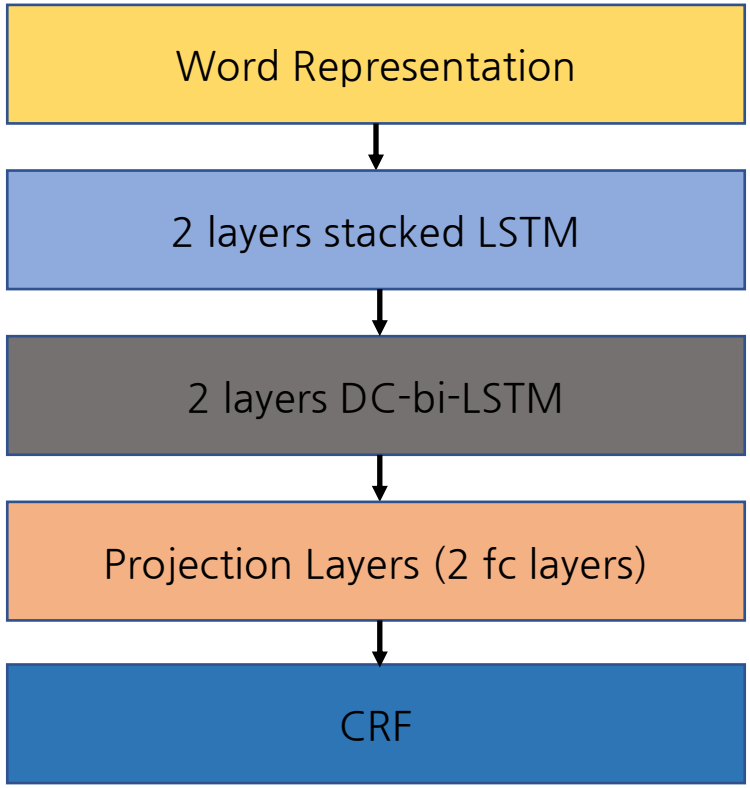
Approach

4. 이제 모델 건드려 보자!

```python
def project_layer(self, lstm_outputs, name=None):
    """
    hidden layer between lstm layer and logits
    :param lstm_outputs: [batch_size, num_steps, emb_size]
    :return: [batch_size, num_steps, num_tags]
    """
    with tf.variable_scope("project" if not name else name, reuse=tf.AUTO_REUSE):
        with tf.variable_scope("hidden", reuse=tf.AUTO_REUSE):
            x = lstm_outputs
            x = tf.layers.dense(x, x.get_shape().as_list()[-1] // 2,
                                kernel_initializer=self.he_uni,
                                kernel_regularizer=self.l2_reg,
                                bias_initializer=tf.zeros_initializer())
            x = tf.nn.leaky_relu(x, alpha=0.2)
            # x = tf.nn.tanh(x)
            # x = gelu(x)

            # x += lstm_outputs

        # project to score of tags
        with tf.variable_scope("logits", reuse=tf.AUTO_REUSE):
            x = tf.layers.dense(x, units=self.num_tags,
                                kernel_initializer=self.he_uni,
                                kernel_regularizer=self.l2_reg,
                                bias_initializer=tf.zeros_initializer())

            pred = tf.reshape(x, (-1, self.word_num_steps, self.num_tags))

        return pred
```

Approach

4. 이제 모델 건드려 보자!



Approach

5. Hyperparameters

batch size : **20**
dropout rate : **0.5**
l2 regularization : 5e-4
LSTM unit size : **256**
character embedding size : **256**

optimizer : Adam
learning rate : 1e-3
learning rate decay factor : 0.9 (1 epoch, exponential decay, stair case)
gradient clip : 5.0

kernel initializer : HE uniform, factor 3.0, mode ‘FAN_AVG’
embedding initializer : HE uniform, factor 3.0, mode ‘FAN_IN’

Result

6. Test F1 Score

Model	F1 Score
baseline	71.2xxx
baseline tuned	73.3xxx
2 layers stacked LSTM w/ reversed sequence	74.43xx
2 layers stacked LSTM w/ reversed sequence + 2 layers DC-bi-LSTM	**74.7695**

https://github.com/kozistr/naver-nlp-challenge-2018 ← 코드는 요기

Result

7. 아쉬운 거

- Attention 류 모델을 테스트 못 해본 거
- LR Scheduling method 도 많이 못 바꿔 본 거
- Session Timeout ㅠㅠ...

Q & A