

# NAVER NLP Challenge 2018

## Semantic Role Labeling (SRL)

**kozistr\_team / HyeongChan Kim**

[kozistr@gmail.com](mailto:kozistr@gmail.com)

ME

A blue square containing the word "kozistr" in white lowercase letters.

kozistr

**KOREATECH** 재학  
**VOYAGERx** Intern  
<http://kozistr.tech>

뭐부터 하지??

# Approach

## 1. Cell 부터 바꿔보자!

LSTMCell (Long Short-Term Memory)

GLSTMCell (Group Long Short-Term Memory)

GRUCell (Gated Recurrent Unit)

NASCell (Neural Architecture Search)

LayerNormBasicLSTMCell

SRUCell (Simple Recurrent Unit)

CoupledInputForgetGateLSTMCell

# Approach

## 1. Cell 부터 바꿔보자!

LSTMCell (Long Short-Term Memory)

GLSTMCell (Group Long Short-Term Memory)

GRUCell (Gated Recurrent Unit)

NASCell (Neural Architecture Search)

LayerNormBasicLSTMCell

SRUCell (Simple Recurrent Unit)

CoupledInputForgetGateLSTMCell

Best!

# 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](#).

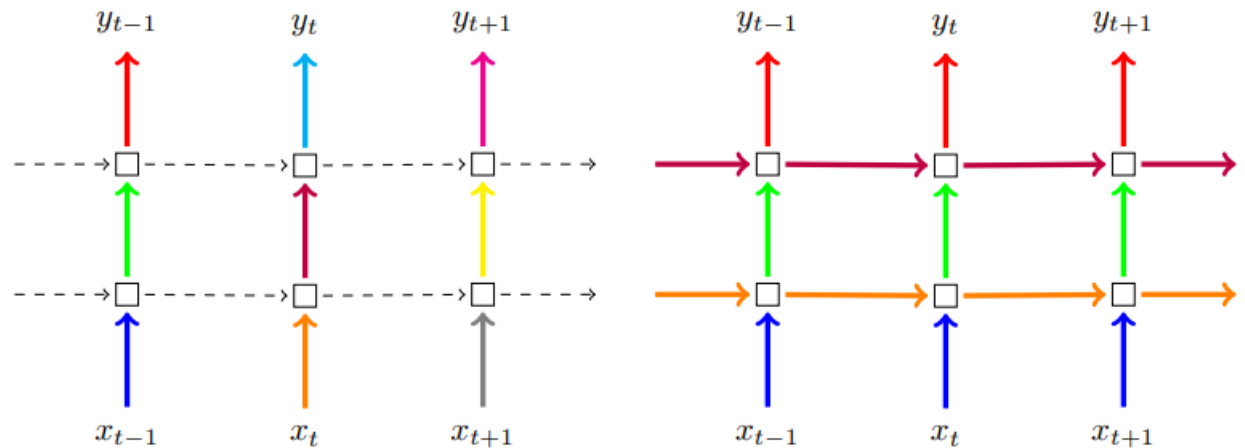
Operator adding dropout to inputs and outputs of the given cell.

<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  
HighwayWrapper  
DropoutWrapper  
ResidualWrapper

CIFGLSTMCell  
HighwayWrapper  
ResidualWrapper  
DropoutWrapper

CIFGLSTMCell  
ResidualWrapper  
HighwayWrapper  
DropoutWrapper

CIFGLSTMCell  
ResidualWrapper  
DropoutWrapper  
HighwayWrapper

# Approach

3. 순서는?!

CIFGLSTMCell  
HighwayWrapper  
DropoutWrapper  
ResidualWrapper

↓  
Best!

CIFGLSTMCell  
HighwayWrapper  
ResidualWrapper  
DropoutWrapper

CIFGLSTMCell  
ResidualWrapper  
HighwayWrapper  
DropoutWrapper

CIFGLSTMCell  
ResidualWrapper  
DropoutWrapper  
HighwayWrapper

CoupledInputForgetGate (CIFGLSTMCell)

# Approach

## 4. 이제 모델 건드려 보자!

The screenshot shows a Google search results page for the query "semantic role labeling in tensorflow". The search bar at the top contains the query. Below the search bar, there are tabs for "전체" (All), "동영상" (Videos), "이미지" (Images), "뉴스" (News), "쇼핑" (Shopping), "더보기" (More), "설정" (Settings), and "도구" (Tools). The search results are displayed below the tabs. The first result is titled "semantic role labeling in tensorflow에 대한 학술자료" (Academic papers on semantic role labeling in tensorflow) and lists three papers: "Allennlp: A deep semantic natural language ..." by Gardner, "Frame-semantic parsing with softmax-margin ..." by Swayamdipta, and "Informed Self-Attention for Semantic Role Labeling" by Strubell. The second result is titled "GitHub - XMUNLP/Tagger: Deep Semantic Role Labeling with Self ..." and provides a link to the GitHub repository. The third result is titled "Deep semantic role labeling using Tensorflow - GitHub" and provides a link to the GitHub repository. The fourth result is titled "Topic: semantic-role-labeling · GitHub" and provides a link to the GitHub topic page. The fifth result is titled "GitHub - luheng/deep\_srl: Code and pre-trained model for: Deep ..." and provides a link to the GitHub repository. The sixth result is titled "GitHub - shengc/tf-lstm-crf-tagger: TensorFlow Implementation For ..." and provides a link to the GitHub repository.

semantic role labeling in tensorflow

전체 동영상 이미지 뉴스 쇼핑 더보기 설정 도구

검색결과 약 70,000개 (0.44초)

**semantic role labeling in tensorflow에 대한 학술자료**  
Allennlp: A deep **semantic** natural language ... - Gardner - 30회 인용  
Frame-**semantic** parsing with softmax-margin ... - Swayamdipta - 14회 인용  
... -Informed Self-Attention for **Semantic Role Labeling** - Strubell - 10회 인용

**GitHub - XMUNLP/Tagger: Deep Semantic Role Labeling with Self ...**  
<https://github.com/XMUNLP/Tagger> 이 페이지 번역하기  
Deep **Semantic Role Labeling** with Self-Attention. Contribute to ... Prerequisites. python2; A newer version of **TensorFlow**; GloVe embeddings and srlconll scripts ...  
이 페이지를 3번 방문했습니다. 최근 방문 날짜: 18. 12. 22

**Deep semantic role labeling using Tensorflow - GitHub**  
<https://github.com/jgung/semantic-role-labeling> 이 페이지 번역하기  
Deep **semantic role labeling** using **Tensorflow**. Contribute to jgung/semantic-role-labeling development by creating an account on GitHub.  
이 페이지를 3번 방문했습니다. 최근 방문 날짜: 18. 12. 5

**Topic: semantic-role-labeling · GitHub**  
<https://github.com/topics/semantic-role-labeling> 이 페이지 번역하기  
Deep **Semantic Role Labeling** with Self-Attention. deep-learning tagging ... **TensorFlow**  
Implementation of deep learning algorithm for NLP. nlp deep-learning ...

**GitHub - luheng/deep\_srl: Code and pre-trained model for: Deep ...**  
[https://github.com/luheng/deep\\_srl](https://github.com/luheng/deep_srl) 이 페이지 번역하기  
@inproceedings{he2017deep, title={Deep **Semantic Role Labeling**: What Works and What's Next}, author={He, Luheng and Lee, Kenton and Lewis, Mike and ...  
이 페이지를 18. 12. 22에 방문했습니다.

**GitHub - shengc/tf-lstm-crf-tagger: TensorFlow Implementation For ...**  
<https://github.com/shengc/tf-lstm-crf-tagger> 이 페이지 번역하기  
**TensorFlow** Implementation For [Neural Architecture for Named Entity ... that has nature in sequence learning, including NER, POS, **Semantic Role Labeling**, etc.

# 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

SessionTimeout

SessionTimeout

SessionTimeout

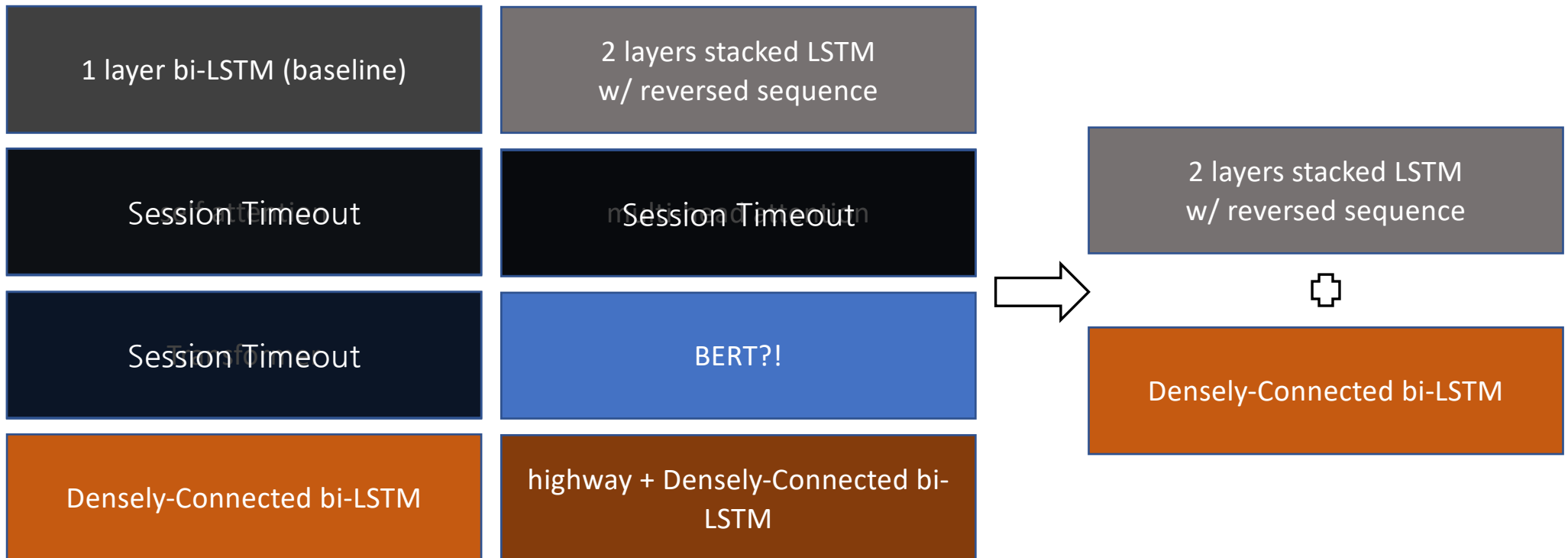
BERT?!

Densely-Connected bi-LSTM

highway + Densely-Connected bi-  
LSTM

# Approach

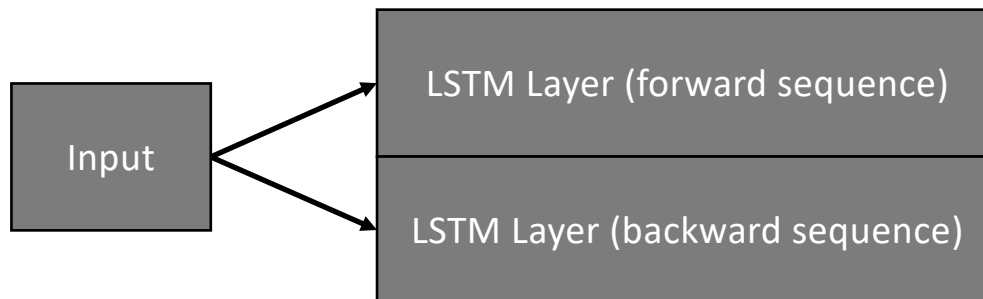
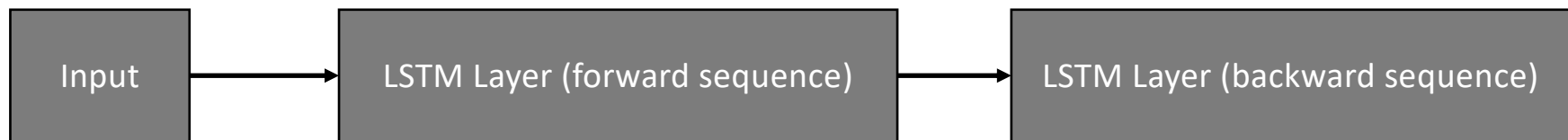
## 4. 이제 모델 건드려 보자!



# Approach

## 4. 이제 모델 건드려 보자!

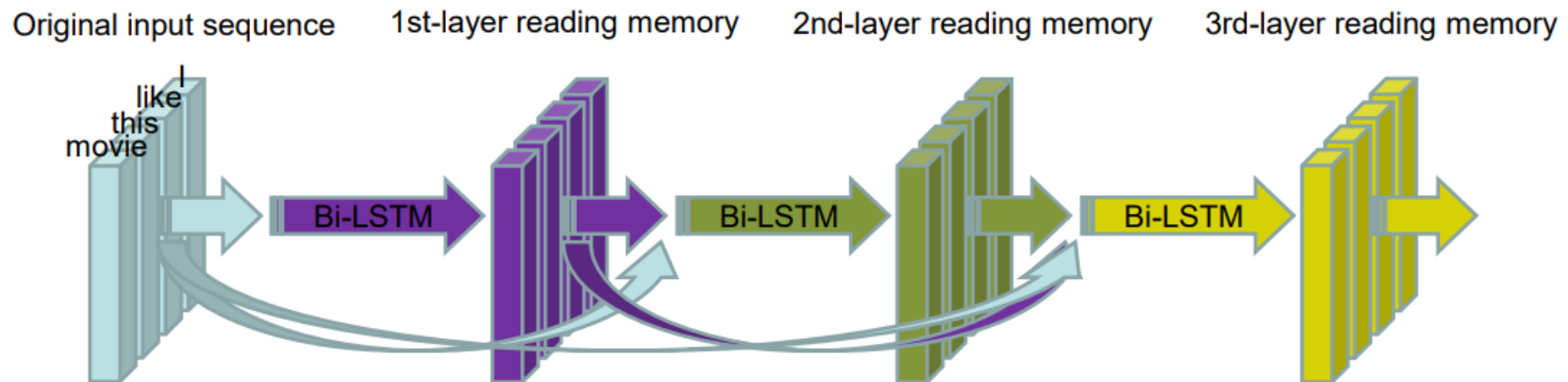
2 layers stacked LSTM w/ reversed sequence (bi-LSTM wisely)



# Approach

## 4. 이제 모델 건드려 보자!

Densely-Connected bi-LSTM



<https://arxiv.org/abs/1802.00889>



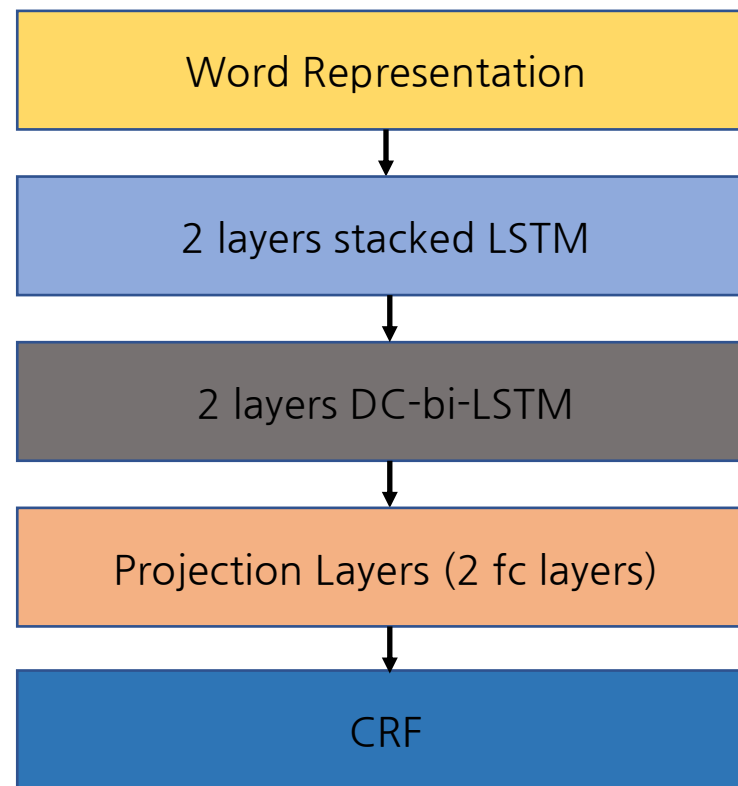
# Approach

## 4. 이제 모델 건드려 보자!

```
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	<b>74.7695</b>

<https://github.com/kozistr/naver-nlp-challenge-2018> ← 코드는 여기

# Result

## 7. 아쉬운 거

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

Q & A