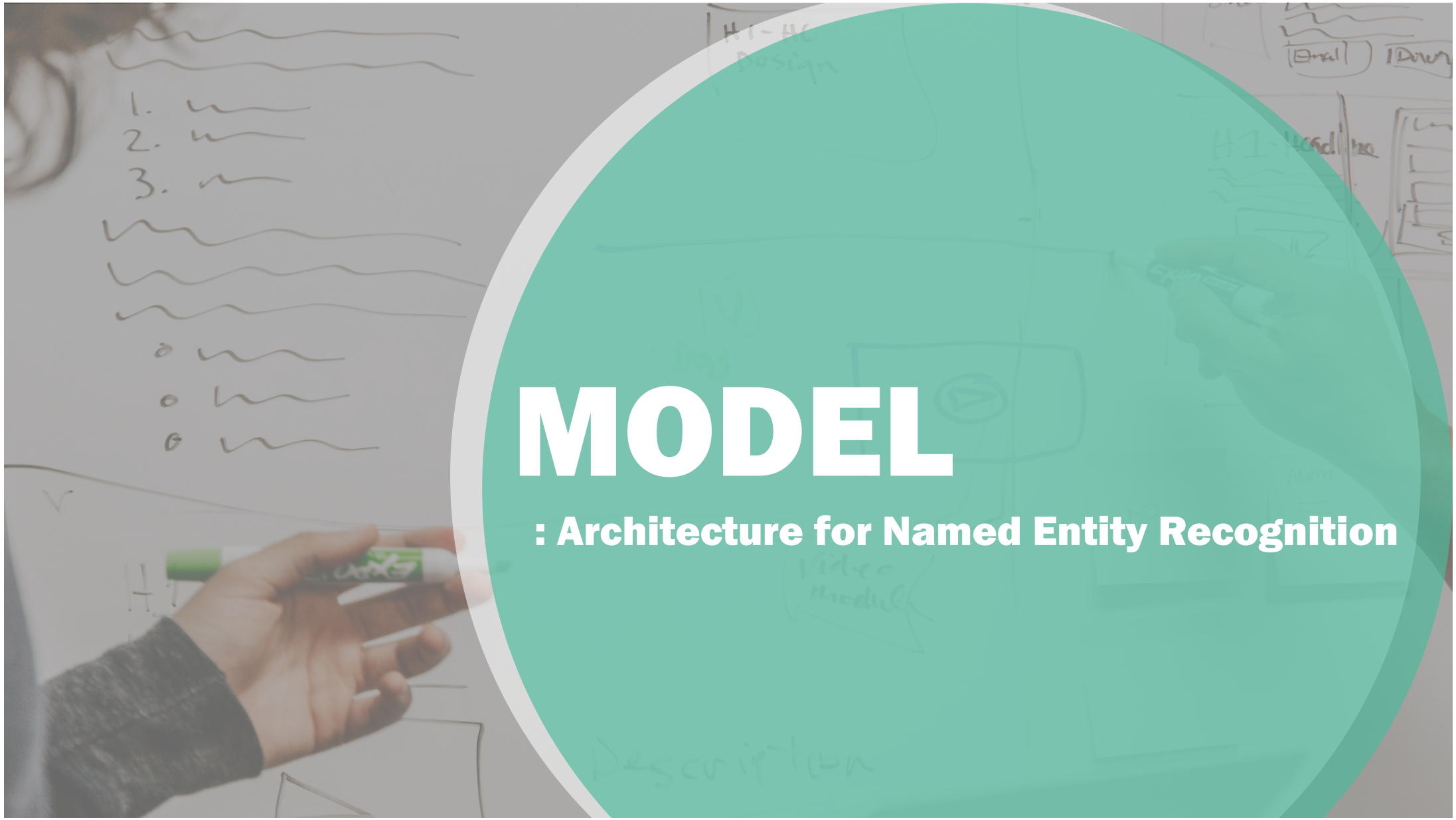
네이버 NLP Challenge 후기

이신의, 박장원, 박종성
연세대학교 데이터공학 연구실

2018.12.28

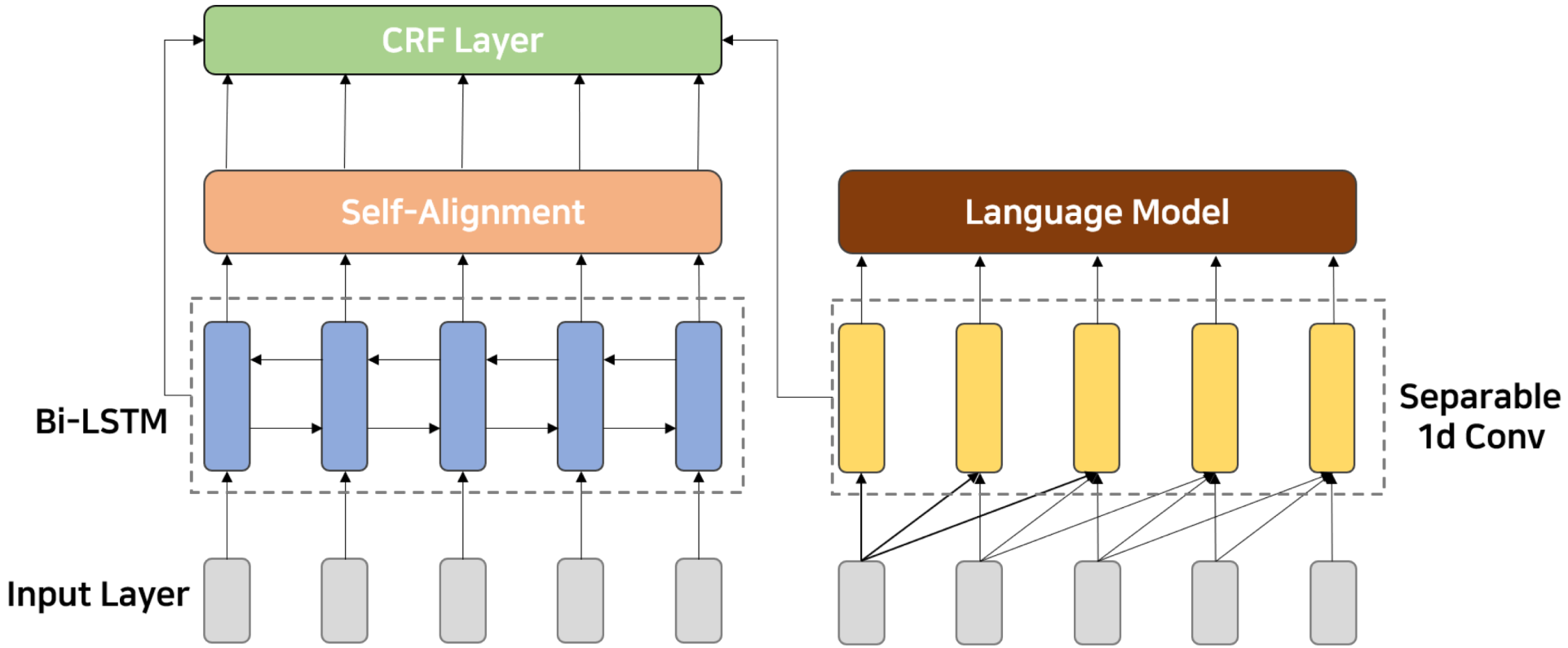
Contents

1. Model
2. Hyperparameter
3. To be considered..


MODEL
: Architecture for Named Entity Recognition

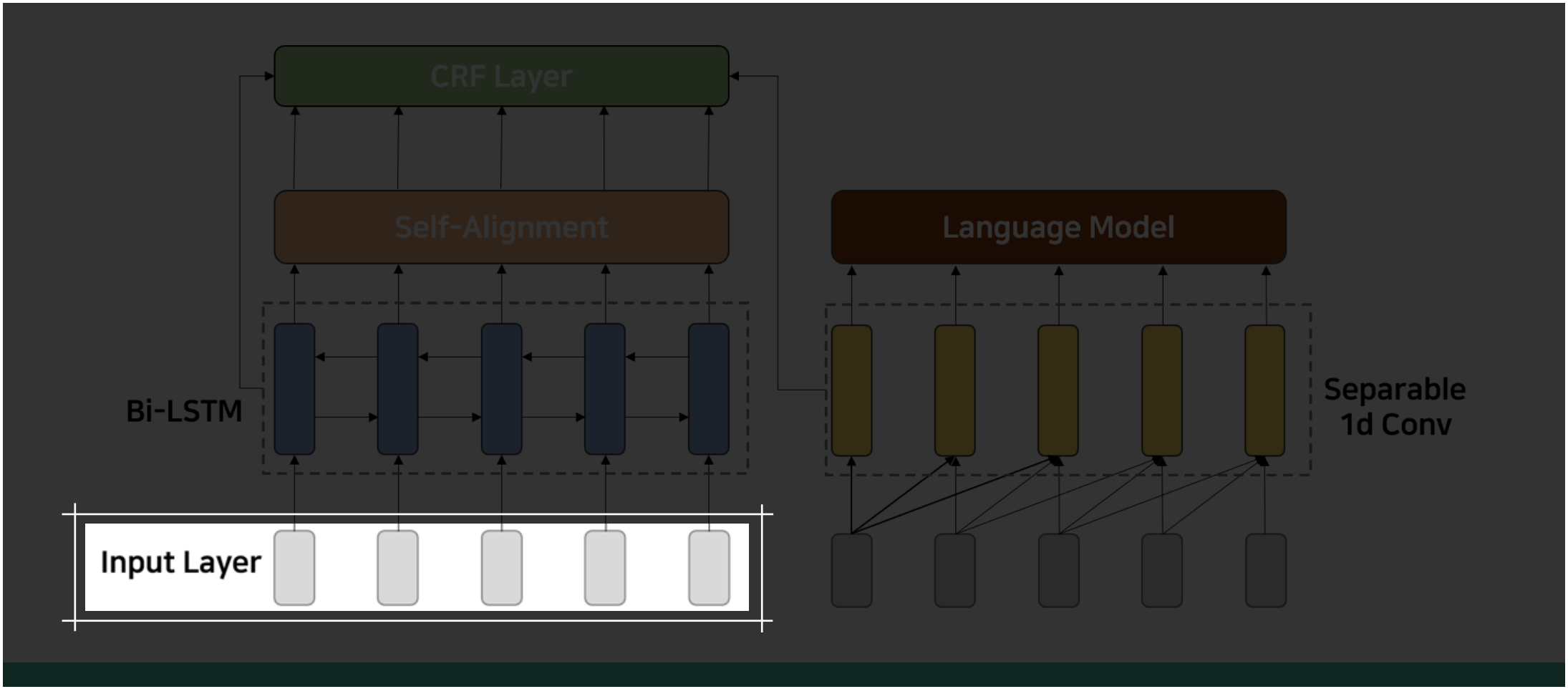
1 Model

Model Architecture



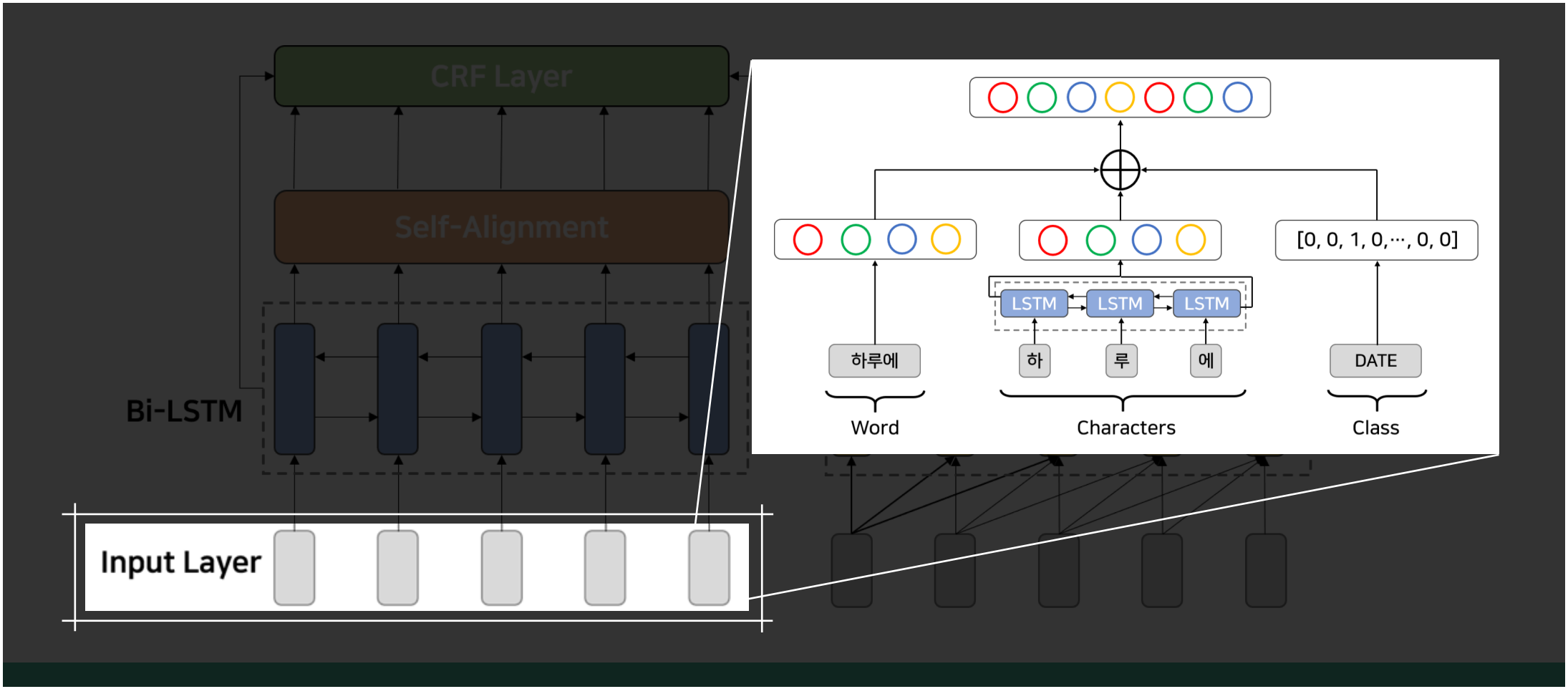
1 Model

1. Input Layer



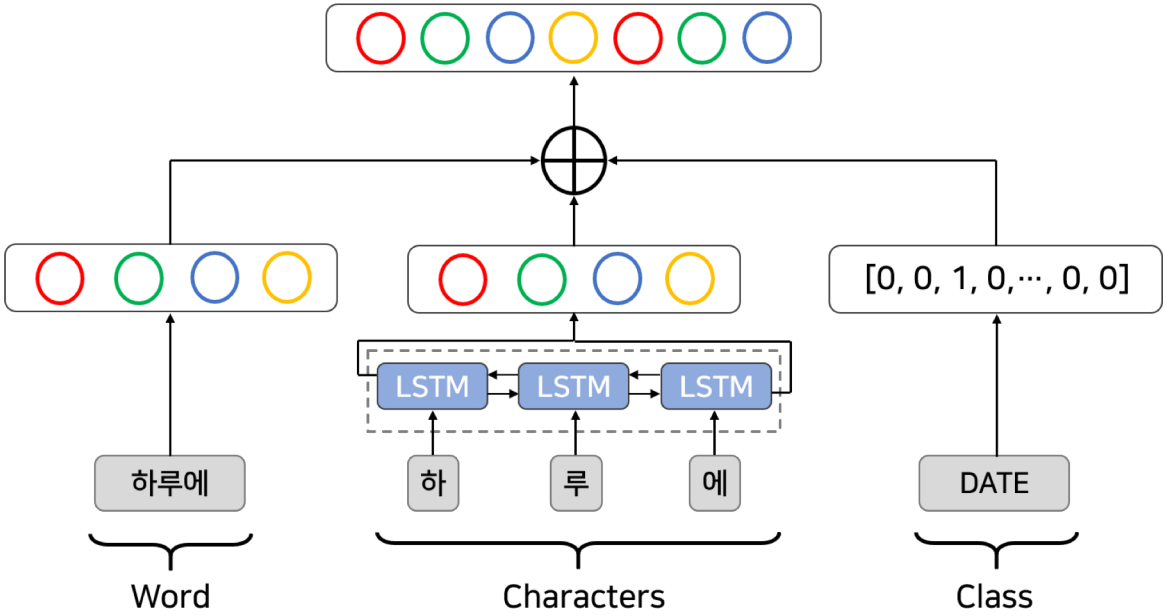
1 Model

1. Input Layer



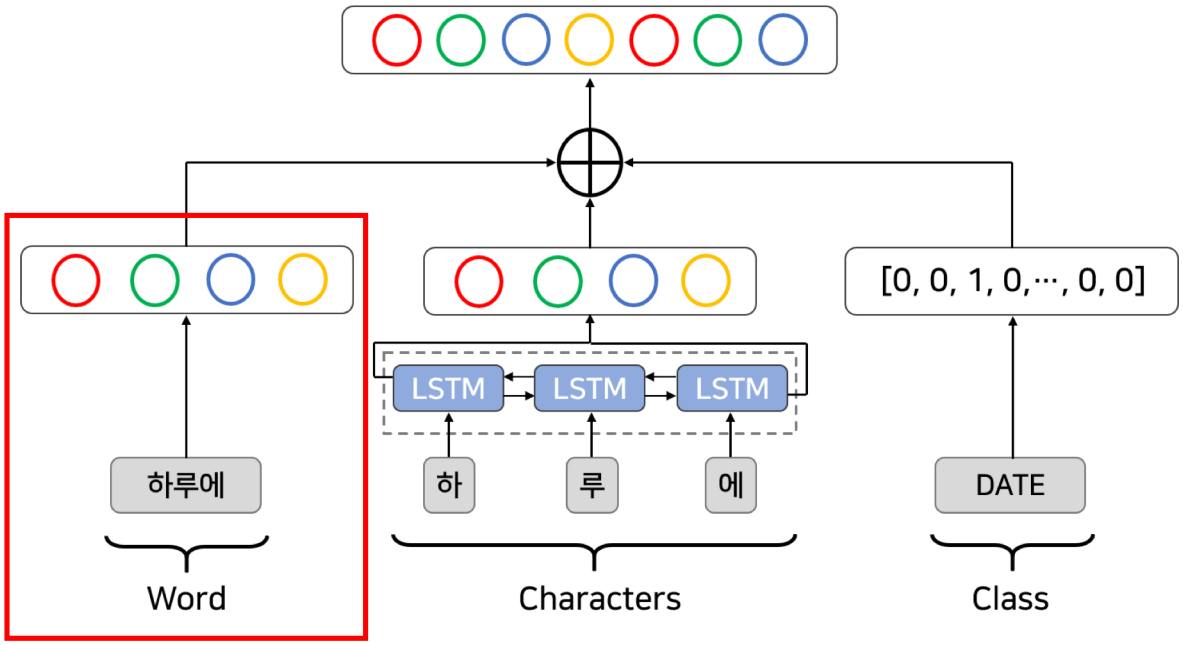
1 Model

1. Input Layer



1 Model

1. Input Layer

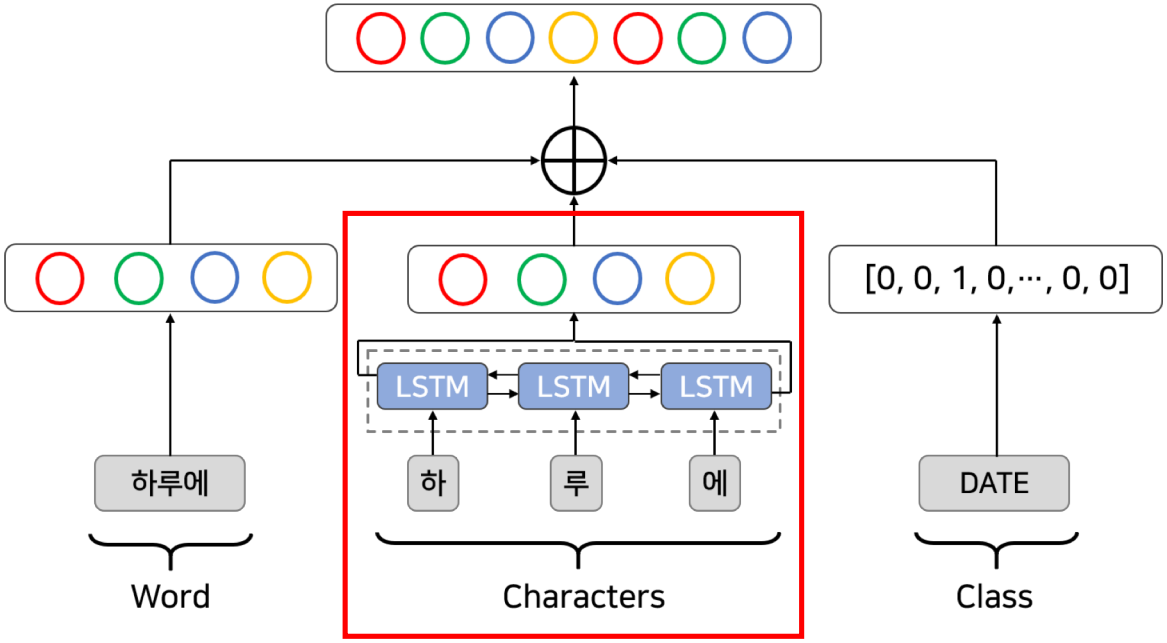


1. 어절 (Word)

- fastText (pretrained word embedding) 이용하여 dense word vector 만듦

1 Model

1. Input Layer

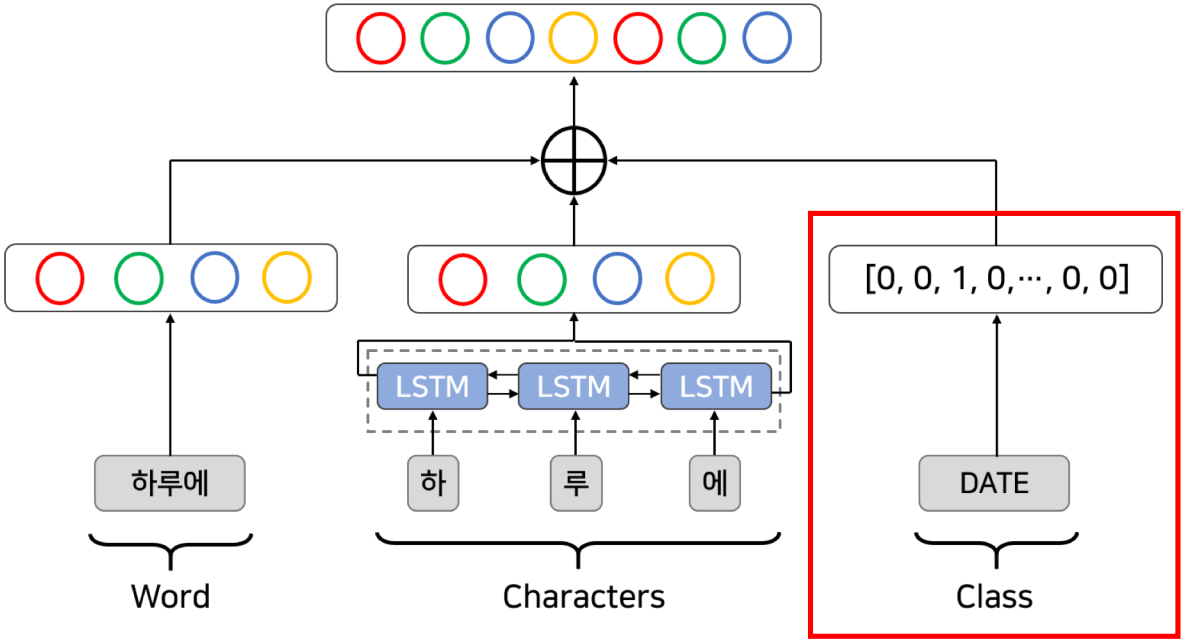


2. 음절 (Character)

- "을", "과" 등 음절 대부분이 fastText word embedding에도 존재
- 동일한 fastText를 이용하여 어절의 각 음절을 dense character vector로 만듦
- Bi-LSTM으로 encoding한 후 last hidden state을 어절의 representation
- 음절 단위의 pretrained embedding이 어절보다 성능 향상에 더 큰 기여!

1 Model

1. Input Layer

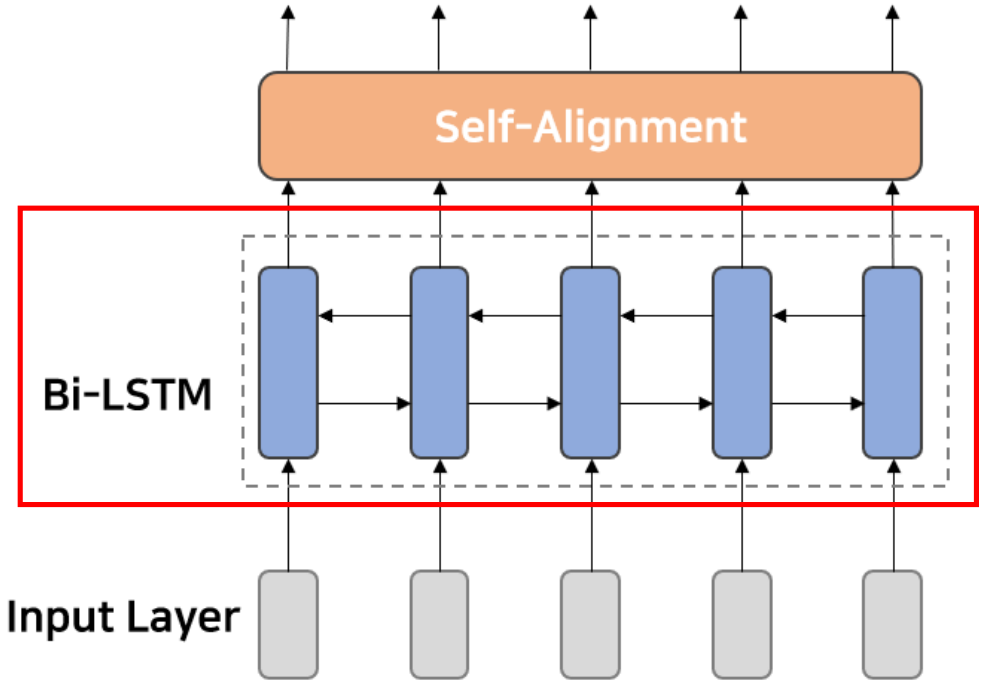


3. 어절 단위의 개체명 one-hot encoding

- 학습 데이터에서 각 어절이 어떠한 개체명에 속하는지 사전으로 저장
- 학습 데이터에서 등장하지 않았던 어절은 "O" class
- 해당 encoding이 있을 때와 없을 때의 성능차가 상당

1 Model

2. Bi-LSTM



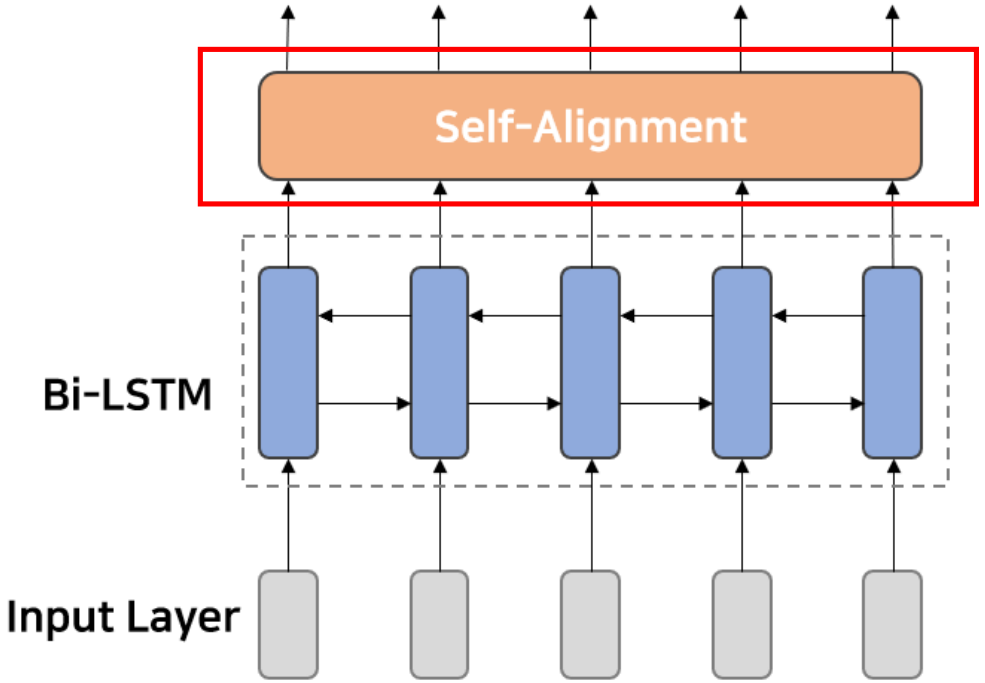
Bi-directional LSTM

- Vanilla LSTM, LayerNorm LSTM, Coupled-Input-Forget-Gate LSTM …
- Time step 마다 layer normalization은 속도 저하
- Coupled Input Forget Gate LSTM
  - 더 적은 수의 parameter, 빠른 속도, 높은 성능

$$f_t = \sigma(W_{xh\_f}x_t + W_{hh\_f}h_{t-1} + b_{h\_f})$$
$$o_t = \sigma(W_{xh\_o}x_t + W_{hh\_o}h_{t-1} + b_{h\_o})$$
$$g_t = tanh(W_{xh\_g}x_t + W_{hh\_g}h_{t-1} + b_{h\_g})$$
$$c_t = f_t \odot c_{t-1} + (1 - f_t) \odot g_t$$
$$h_t = o_t \odot \tanh(c_t)$$

1 Model

3. Self-Attention



Attention

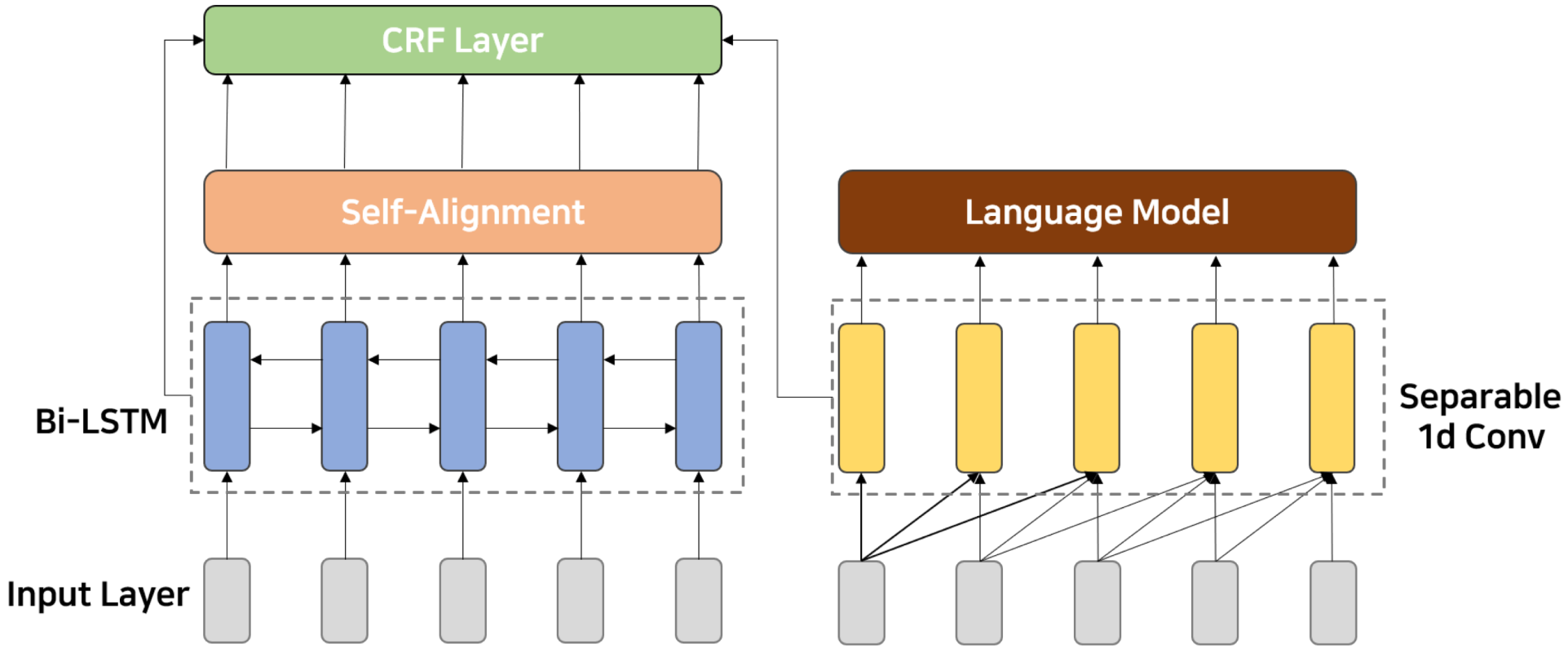
- Bi-LSTM의 hidden states간의 self-alignment
- 자기 자신과의 attention score는 -∞로 지정
- Multi-head dot product attention like Transformer

- $Attention(Q,K,V) = softmax\left(\frac{QK^T}{\sqrt{d}}\right)V$

  $(Q_i^T K_j := -\infty \text{ if i} = \text{j})$
- $MultiHead(Q,K,V) = Concat(head_1, \dots, head_8)W^O$
- $head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right)$
- $Q,K,V :=$ hidden states of Bi-LSTM

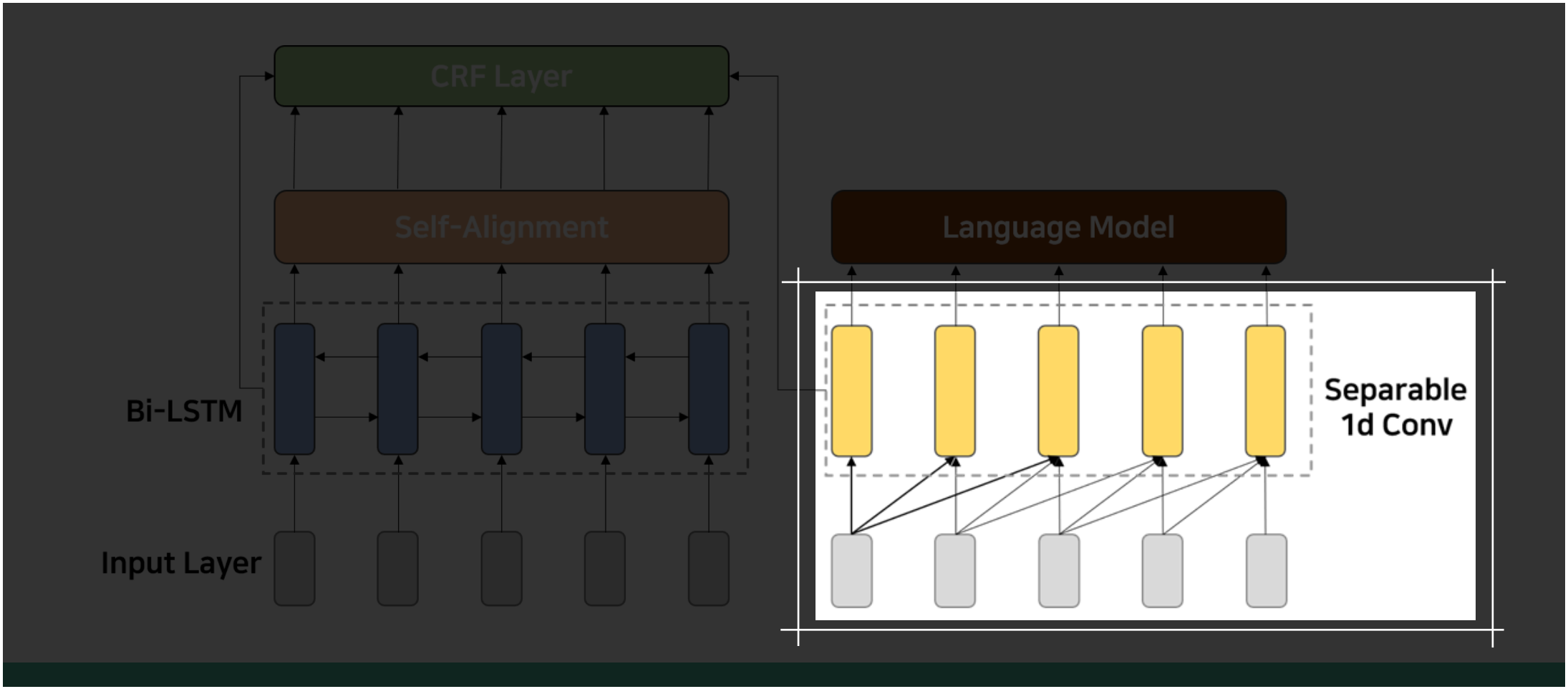
1 Model

4. Masked Separable Conv1d



1 Model

4. Masked Separable Conv1d



1 Model

4. Masked Separable Conv1d

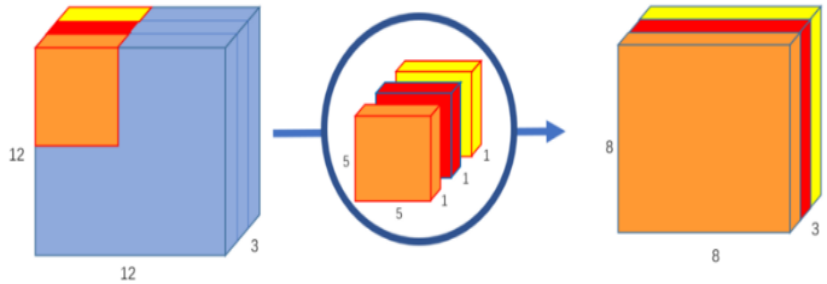


Image 6: Depthwise convolution, uses 3 kernels to transform a 12x12x3 image to a 8x8x3 image

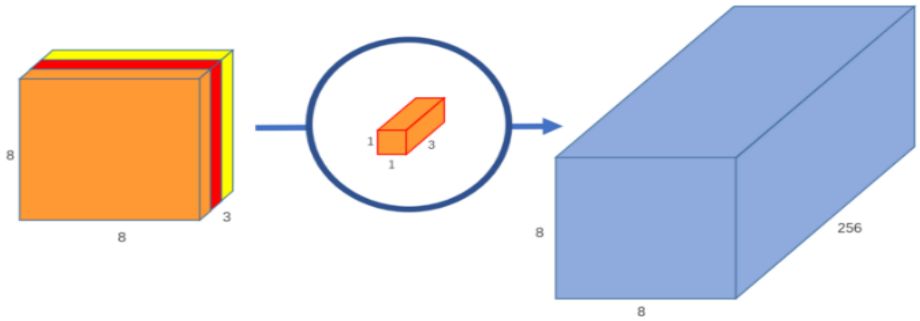
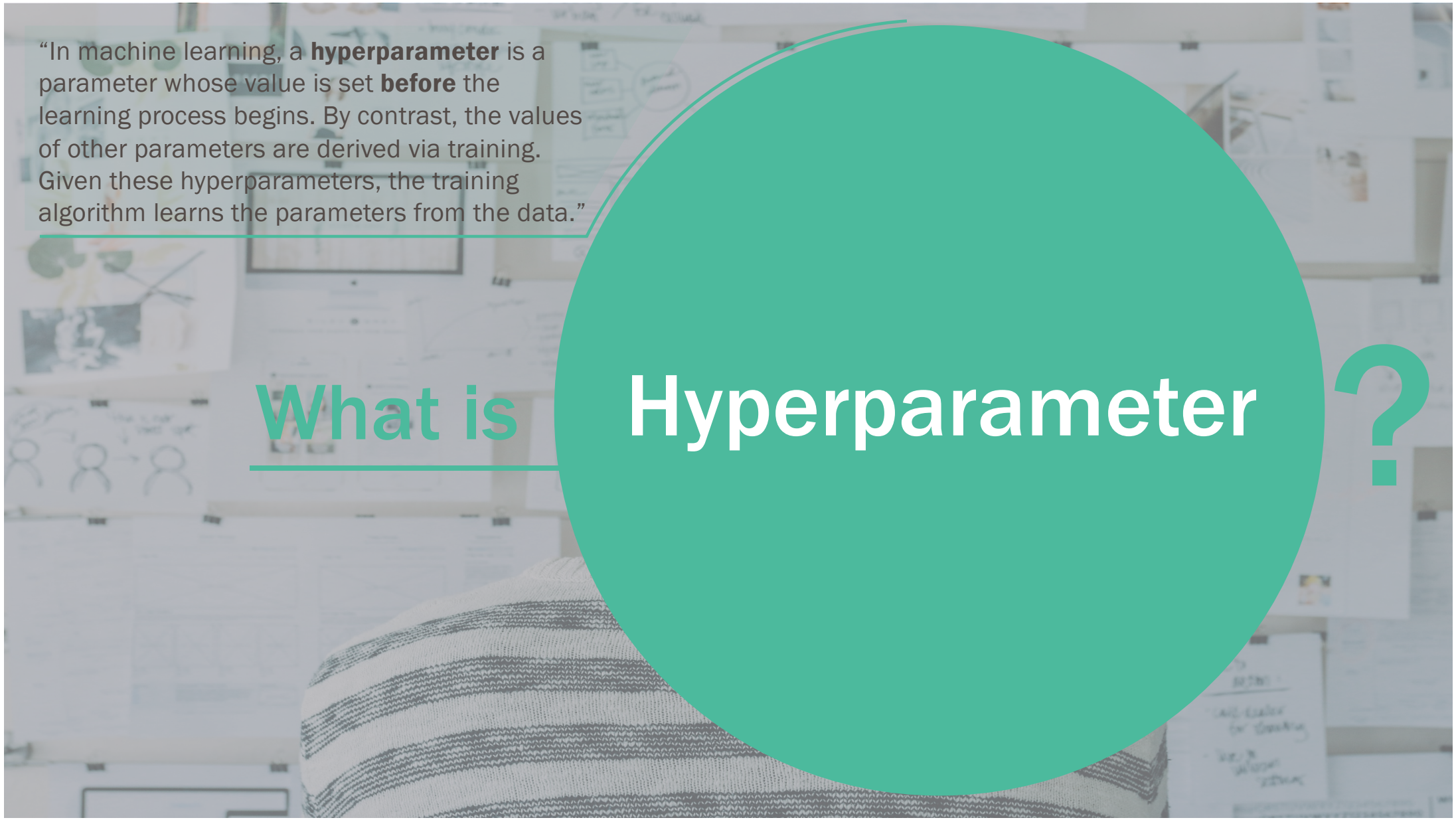


Image 8: Pointwise convolution with 256 kernels, outputting an image with 256 channels

Pictures from https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

Masked Separable Conv1d
for richer representation

- Masking for autoregressive
  - (kernel size -1) 만큼 zero padding 하여 현재 time step 보다 앞선 token을 보지 못하게 함
- Depth-wise convolution
  - 각 channel마다 convolution
- Pointwise Convolution
  - 1x1 convolution을 이용하여 channel interaction
- Positional embedding
  - convolution 연산 후, Sinusoid embedding (like Transformer)을 더함
- 각 kernel 별로 나온 feature map을 모두 concatenate


“In machine learning, a **hyperparameter** is a parameter whose value is set **before** the learning process begins. By contrast, the values of other parameters are derived via training. Given these hyperparameters, the training algorithm learns the parameters from the data.”
What is
Hyperparameter
?

2 Hyperparameter

연금술? 흑마법?

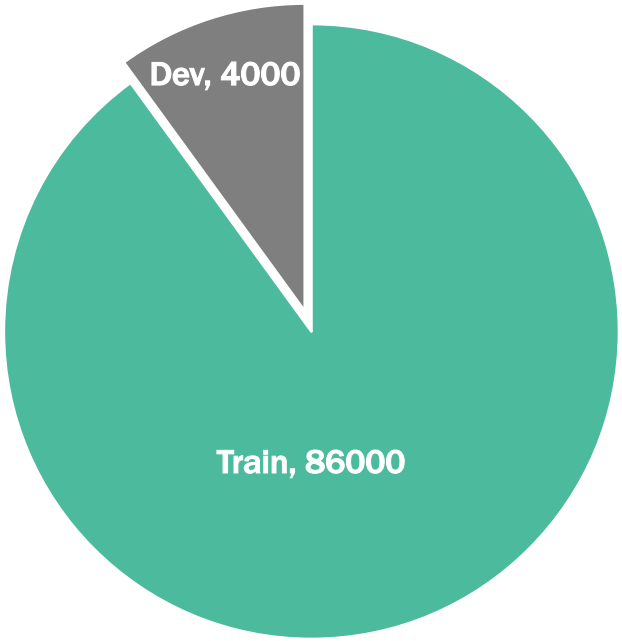
- Model만으로 F1 88.92%에 도달
- F1 89.72%까지 약 0.8%는 Hyperparameter Tuning (a.k.a. 연금술)로 달성
- 모델 제작에 2일, 연금술에 7일 소요...

“There’s a place for alchemy. Alchemy worked.”

- Ali Rahimi

2 Hyperparameter

1. Train-Dev split / Learning Rate Decay



Train-Dev Split

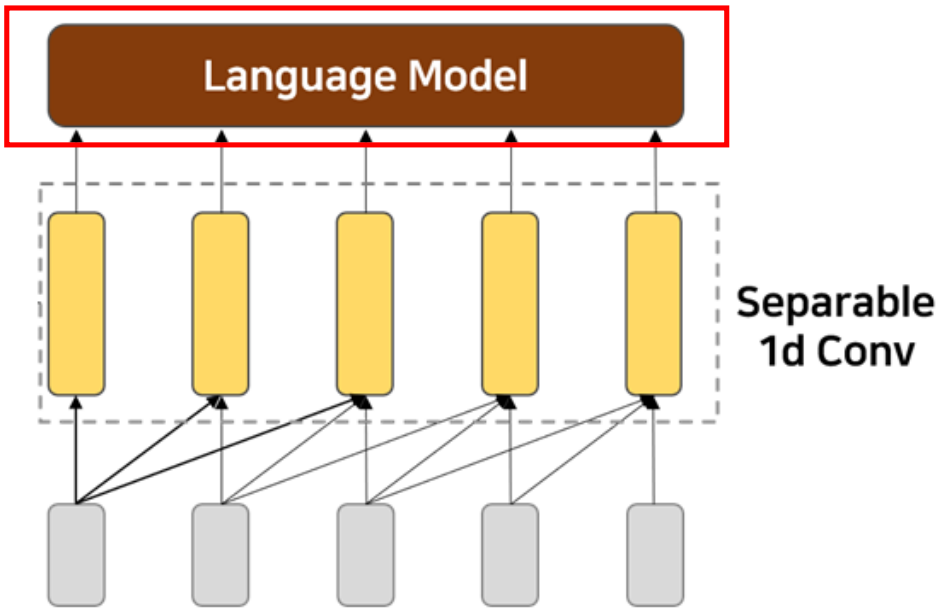
- Train: Dev = 86000 : 4000
- Dev F1 score와 Test F1 score가 비례함 (0.5% 정도 차이 존재)

Learning rate scheduling

- Adam을 사용함에도 learning rate 조절 필요
  - Dev F1 score 기준, 두 번째 epoch부터 성능이 떨어짐
- Learning rate decay를 사용!
  - Epoch이 지날 때 마다 lr에 0.1 곱함
- 세번째 epoch부터는 lr를 더 줄여도 F1이 오르지 않았음....

2 Hyperparameter

2. Auxiliary Loss

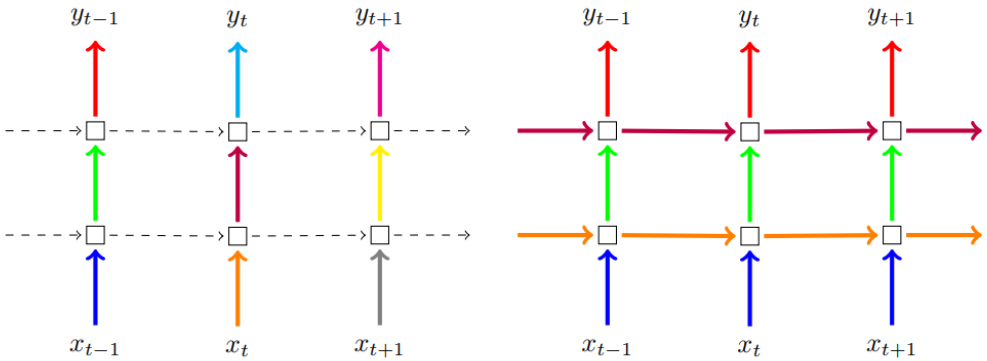


Language Model Loss

- Language Model을 이용하여 representation learning을 하면 down stream task 성능 향상 (BERT, ELMO)
- 외부 데이터를 사용하지 않고 richer representation 을 만드는 것이 목표
- Convolution의 feature map이 NER과 Language Model에 모두 사용
- 앞의 m 어절이 주어지고 다음 어절 예측하는 LM
- Language model loss=$\sum_{k=1}^{n} -\log(t_k \mid t_{k-m}, \ldots, t_{k-1}\ ;\ \theta)$
- Loss = Original loss + 0.5 * Language Model loss

2 Hyperparameter

3. Dropout



(a) Naive dropout RNN (b) Variational RNN

Dropout

- 기본적인 dropout은 25%
- LSTM은 variational dropout으로 10%만
  - Dropout mask를 모든 time step에서 공유
- Dropout은 "적절한" 위치에 적용
  - Embedding 또는 FC layer 이후에는 dropout이 역효과
  - Multi-head attention과 FC layer 직전에 사용

2 Hyperparameter

4. Other Options

Exponential Moving Average

- $W_{t+1} \leftarrow 0.999W_t + 0.001W_{t+1}$

Regularization

- Weight에 L2 Regularization 적용
- L2 Weight Decay = 1e-7

Batch size

- Batch size = [8, 16, 32, 64]
- Batch size를 8로 했을 때 미세하지만 성능이 가장 좋았음

To Be Considered

: Things that we couldn’t use...

3 To be considered

사용하지 못해 아쉬웠던 것들

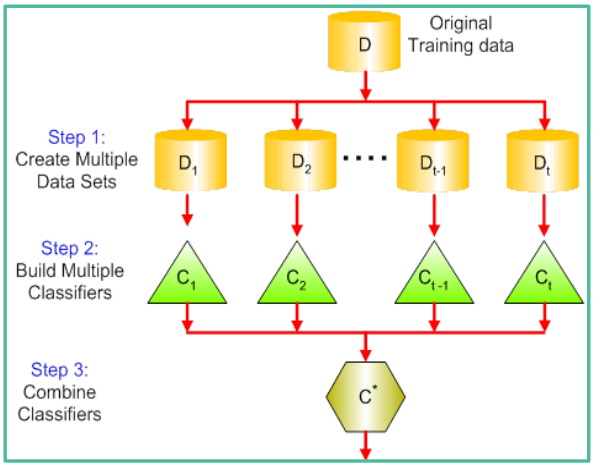
1. Multi-lingual BERT

- 어절 단위 tokenization X
- 마지막 layer에 attention을 이용하여 feature 추출

→ 오히려 성능이 저하, 어절과 sub-word간의 alignment 필요

2. ELMO

- 한국어 데이터셋으로 학습한 모델을 찾을 수 없었음
- 처음부터 ELMO embedding을 학습하기에는 시간 부족



3. Ensemble

- 크레딧 부족으로 적용해보지 못 함

Thank You

Reference

[1] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. ArXiv e-prints.

[2] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word prepresentations. In Proceedings of NAACL, 2018.

[3] A. Howard. Mobilenets: Efficient convolutional neural networks for mobile vision applications. Forthcoming

[4] Yarin Gal. A theoretically grounded application of dropout in recurrent neural networks. arXiv preprint arXiv:1512.05287, 2015.

[5] K. Greff, R. K. Srivastava, J. Koutn´ık, B. R. Steunebrink, and J. Schmidhuber. LSTM: A search space odyssey. CoRR, abs/1503.04069, 2015.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. In Neural Information Processing Systems (NIPS), 2017.

[7] https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728