End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF

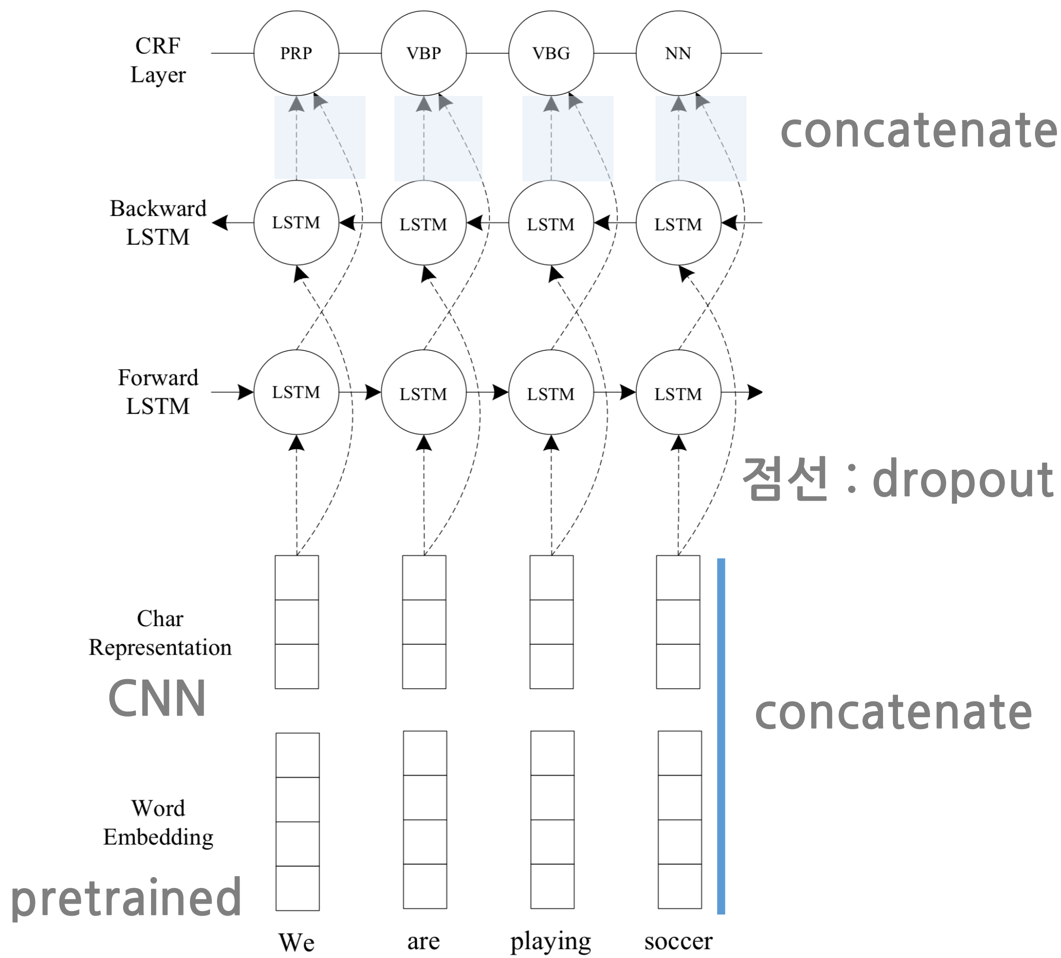
발표자 : 이기창
https://ratsgo.github.io

Introduction

- HMM, CRF 등 기존 모델들은 수작업 피처(hand-crafted features)에 의존
- Character-level CNN으로 뽑은 피처와 Word embedding을 결합
- 이들 임베딩을 Bi-directional LSTM 레이어에 넣고 여기에 Conditional Random Fields 적용
- 피처 엔지니어링 없이 end-to-end 모델을 구축했다는 점 어필

Architecture

- overall

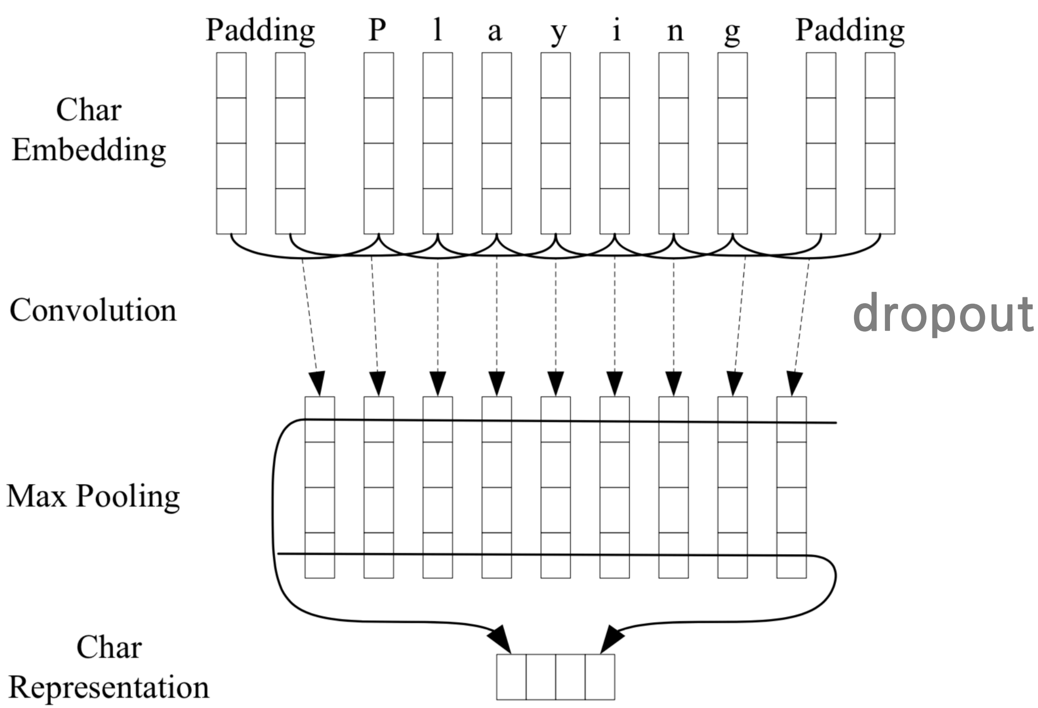


Architecture

- Convolutional Neural Networks (character encoding)

  an effective approach to extract morphological information.

  dropout is applied before character embeddings are input to CNN



Architecture

- BLSTM

  the LSTM's hidden state h_t takes information only from past,
  knowing nothing about the future …
  Then the (forwards and backwards) two hidden states are concatenated
  to form the final output.

Architecture

- Conditional Random Fields + Viterbi Decoding

  For sequence labeling tasks, it is beneficial to consider the correlations between labels in neighborhoods and jointly decode the best chain of labels for a given input sentence.

Conditional Random Fields

- Conditional Probability

$$p(y|x) = \frac{p(x,y)}{p(x)}$$

- Inference (discriminative)

$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|x)$$

Conditional Random Fields

- Conditional Probability

$$p(y|x) = \frac{p(x,y)}{p(x)}$$

- Inference (generative)

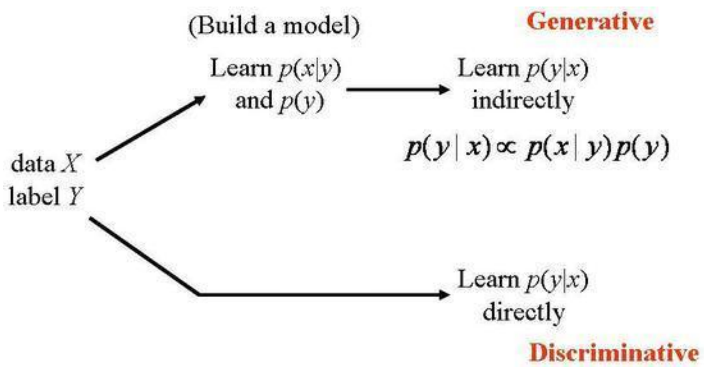
$$\hat{y} = \underset{y}{\operatorname{argmax}}\, p(y|x)$$

$$= \underset{y}{\operatorname{argmax}} \frac{p(x|y) \cdot p(y)}{\boxed{p(x)}}$$ Argmax에 영향을 끼치지 않음

$$= \underset{y}{\operatorname{argmax}}\, p(x|y) \cdot p(y)$$

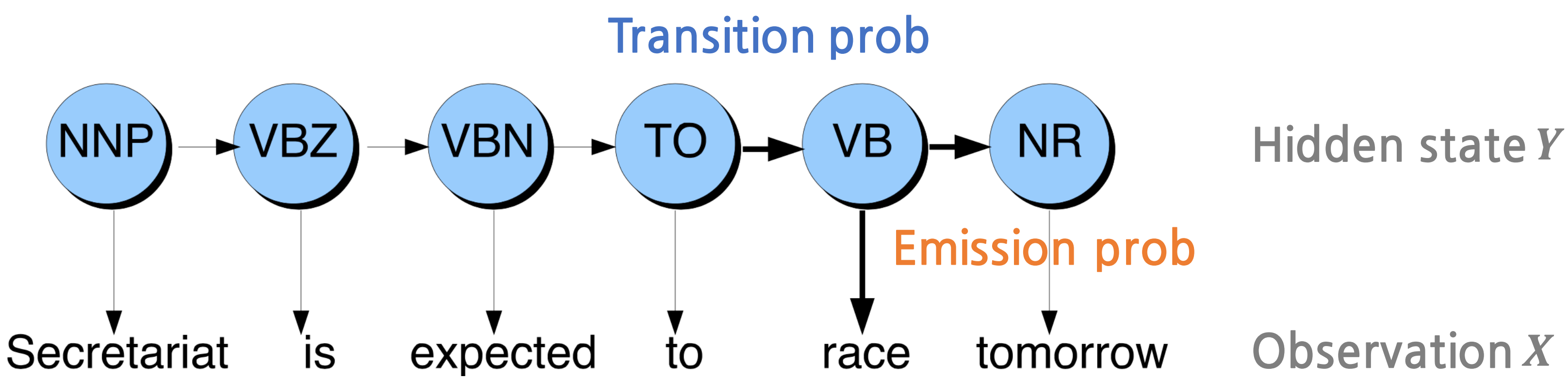
Conditional Random Fields

- Discriminative vs Generative models



Conditional Random Fields

- Hidden Markov Models

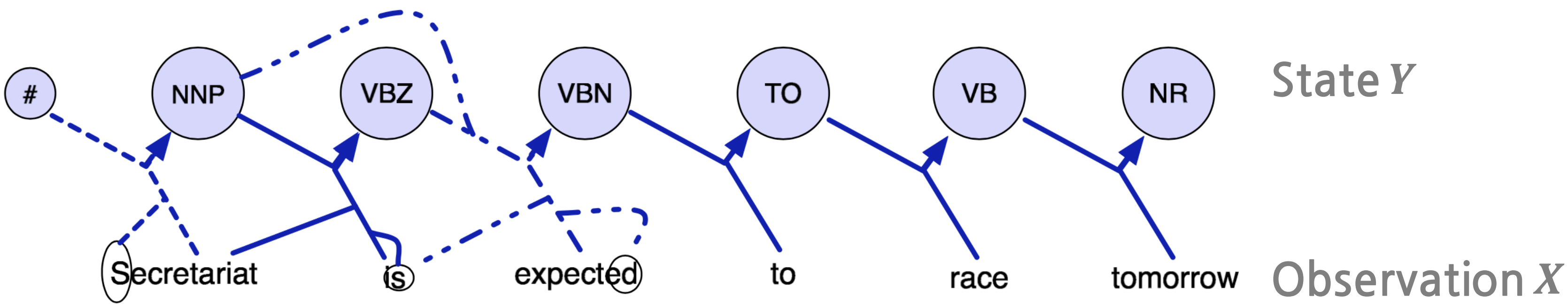


$$\begin{aligned}\hat{Y} &= \underset{Y}{\operatorname{argmax}}\, p(Y|X) \\ &= \underset{Y}{\operatorname{argmax}}\, p(X|Y) \cdot p(Y) \\ &= \underset{Y}{\operatorname{argmax}} \prod_i p(X_i|Y_i) \prod_i p(Y_i|Y_{i-1})\end{aligned}$$

현재 hidden state는 직전 상태에만 의존
현재 관측치는 현재 hidden state에만 의존
Generative model

Conditional Random Fields

- Maximum Entropy Markov Models

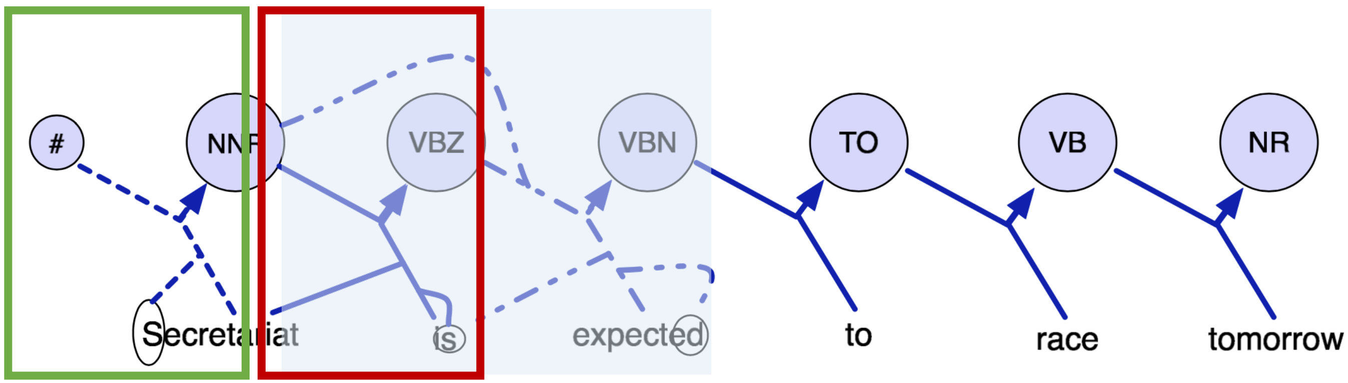


$$
\begin{aligned}
\hat{Y} &= \underset{Y}{\operatorname{argmax}}\, p(Y|X) \\
&= \underset{Y}{\operatorname{argmax}} \prod_i p(Y_i|X_i, Y_{i-1}) \\
&= \underset{Y}{\operatorname{argmax}} \prod_i \boxed{\frac{\exp(\overrightarrow{w_y}^T \overrightarrow{f(X_i, Y_{i-1})})}{\sum_{y' \in Label} \exp(\overrightarrow{w_{y'}}^T \overrightarrow{f(X_i, Y_{i-1})})}}
\end{aligned}
$$

현재 state는 직전 상태에만 의존
시퀀스 추정에 **다양한 자질 활용** (수작업 구축)
시퀀스 추정에 **다항로지스틱 회귀** 적용
Discriminative model

Conditional Random Fields

- Feature?



- 수작업 구축…노가다!

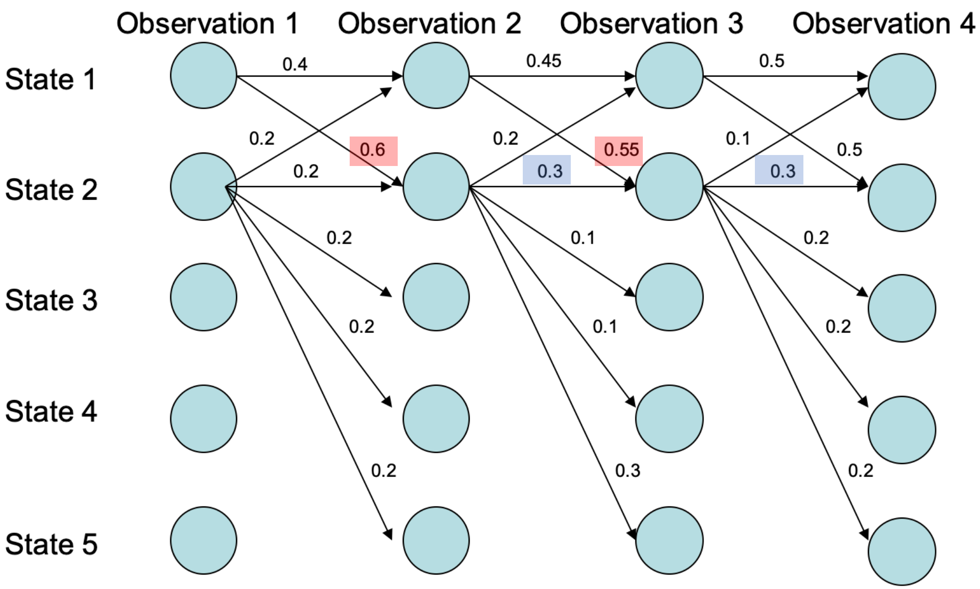
Conditional Random Fields

- CRF : MEMM + Label Bias 해결을 위한 Global normalize

Conditional Random Fields

- Label Bias

  preference of states with lower number of transition over others

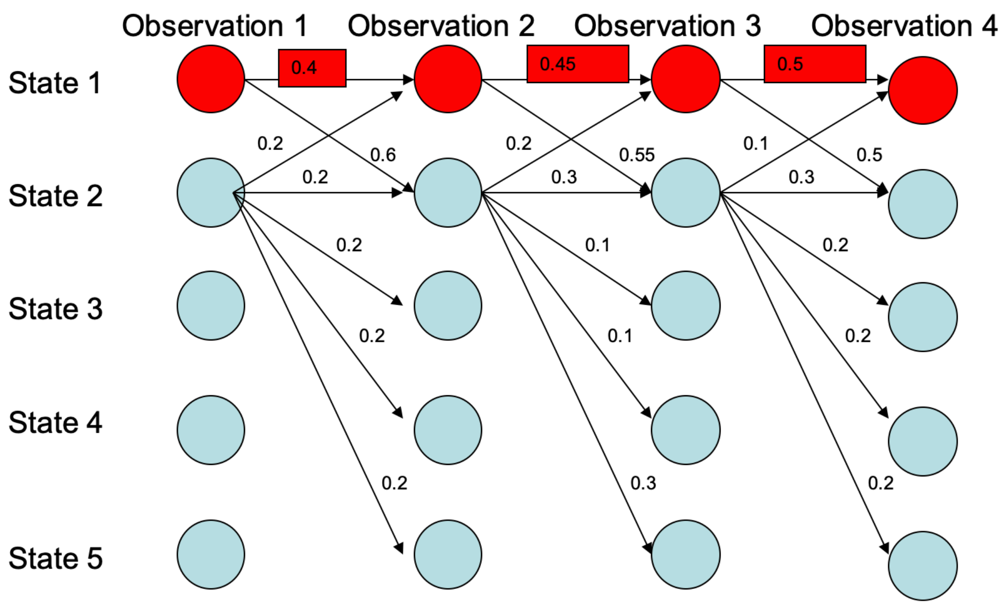


관찰
State1에선 State2로 가려는 경향
State2에선 State2에 남으려는 경향

Conditional Random Fields

- Label Bias

preference of states with lower number of transition over others



Most likely path
1>1>1>1 = 0.4*0.45*0.5=0.09

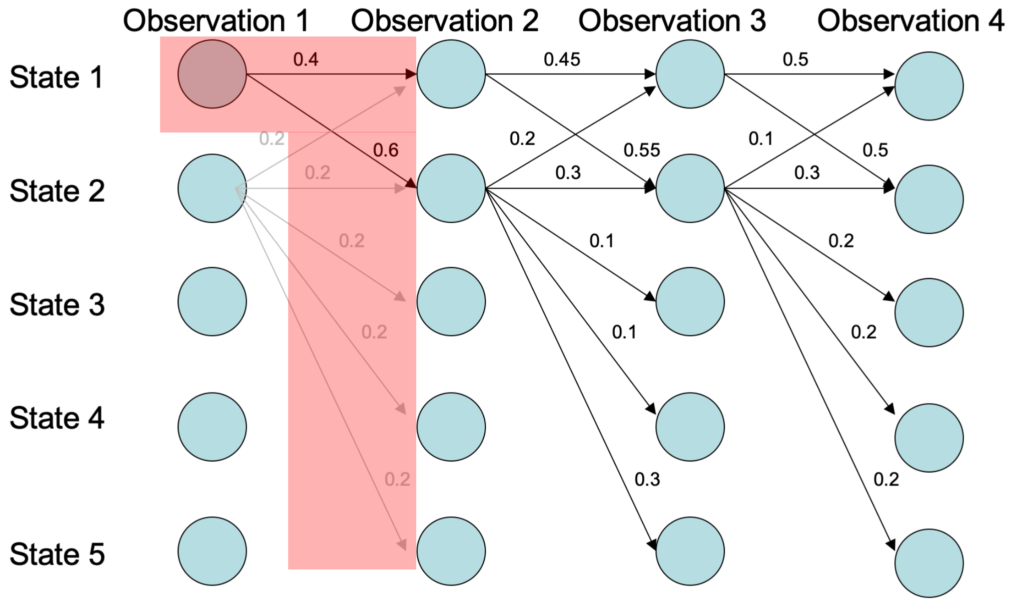
Other paths
1>1>2>2 = 0.4*0.55*0.3=0.066
1>2>1>2 = 0.6*0.2*0.5=0.06
2>2>2>2 = 0.2*0.3*0.3=0.018

Label Bias
transition 가짓 수가 적은 State1으로 label inference를 하려는 현상

Conditional Random Fields

- Solution : 전체 관측치/state에 대한 Global normalize

states with lower number transitions do not have an unfair advantage



Local normalize

(1) 분자 : Observation1이 State1 일 때 각 state로 전이할 스코어

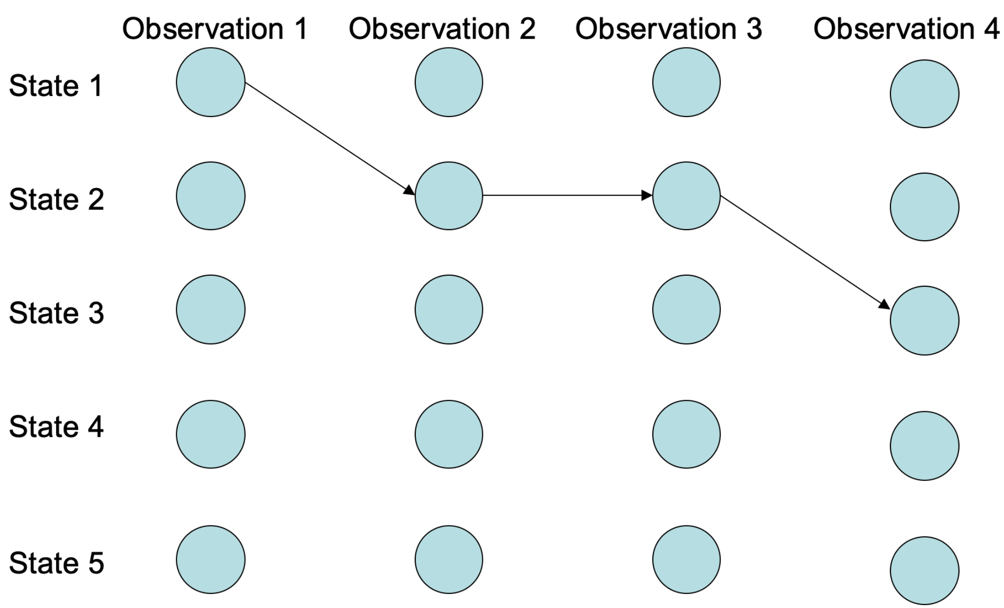
(2) 분모 : Observation1이 각 State로 전이할 경우의 수는 state 종류 수, 즉 레이블 가짓수이며, 이들 모든 스코어의 합

(3) MEMM은 $P(Y_i|X_i,Y_{i-1})$ 계산 (time step i별로 확률 assign)

Conditional Random Fields

- Solution : 전체 관측치/state에 대한 Global normalize

  states with lower number transitions do not have an unfair advantage

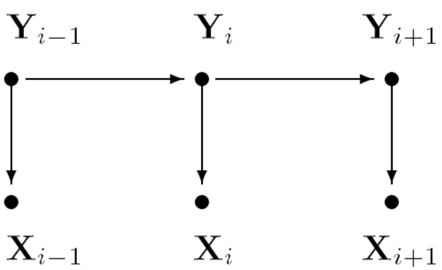


Global normalize
(1) 분자 : Observation1~4의 State sequence가 1>2>2>3일 스코어
(2) 분모 : Observation1~4가 가질 수 있는 state sequence 경우의 수는 5^4가지이며 각각의 스코어의 합
(3) CRF는 $P(Y_{1:n}|X_{1:n})$ 계산
(시퀀스 전체에 확률 assign)

Conditional Random Fields

- 세 기법 비교

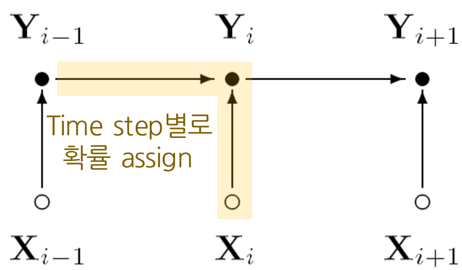
HMM



현재 state는 직전 상태에 의존
현재 관측치는 현재 state에 의존
Generative model

$$\hat{Y} = \arg\max_{Y} P(Y|X)$$
$$= \arg\max_{Y} \prod_i P(X_i|Y_i) \prod_i P(Y_i|Y_{i-1})$$

MEMM



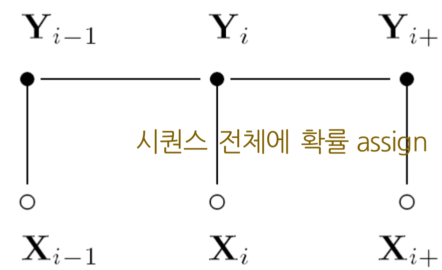
현재 state는 직전 상태에 의존
피처 구축시 다양한 자질 활용 (수작업)
state 예측에 다항 로지스틱 적용
Discriminative model

$$\hat{Y} = \arg\max_{Y} P(Y|X)$$
$$= \arg\max_{Y} \prod_i P(Y_i|X_i, Y_{i-1})$$
$$= \arg\max_{Y} \boxed{\prod_i \frac{s(Y_i|X_i, Y_{i-1})}{\sum_{Y' \in \mathcal{L}} s(Y'_i|X_i, Y'_{i-1})}}$$

Time step별로 확률 assign

분자
직전 레이블이 Y_{i-1}일 때 X_i의 레이블이 Y_i일 스코어
분모
Y_{i-1}에서 Y_i으로 전이할 수 있는 모든 경우 ($\mathcal{L}$=레이블 종류)의 수에 해당하는 스코어 합

CRF



현재 state는 직전 상태에 의존
피처 구축시 다양한 자질 활용 (수작업)
state 예측에 다항 로지스틱 적용
state sequence 확률을 global normalize
Discriminative model

$$\hat{Y} = \arg\max_{Y} P(Y|X)$$
$$= \arg\max_{Y} \frac{\prod_i s(Y_i|X_i, Y_{i-1})}{\sum_{Y' \in \psi} \prod_i s(Y'_i|X_i, Y'_{i-1})}$$

시퀀스 전체에 확률 assign

분자
X_{i-1}, X_i, X_{i+1}의 시퀀스 레이블이 Y_{i-1}, Y_i, Y_{i+1}일 스코어
분모
X_{i-1}, X_i, X_{i+1}가 가질 수 있는 모든 경우의 시퀀스(ψ) 에 해당하는 스코어의 합

Conditional Random Fields

- CRF with deep learning (논문에 나온 수식)

CRF with Deep Learning
: No feature engineering

Markov Assumption

Multinomial Logistic

$$\boxed{p(\boldsymbol{y}|\mathbf{z};}\,\mathbf{W},\mathbf{b}) = \frac{\prod_{i=1}^{n}\psi_i(y_{i-1},y_i,\mathbf{z})}{\sum_{y'\in\mathcal{Y}(\mathbf{z})}\prod_{i=1}^{n}\psi_i(y'_{i-1},y'_i,\mathbf{z})}$$

Global normalize

Discriminative model
: Learn directly $p(y|z)$

$$\text{where } \psi_i(y',y,\mathbf{z}) = \exp(\mathbf{W}^T_{y',y}\mathbf{z}_i + \mathbf{b}_{y',y})$$

Unary Scores

Binary Scores

Conditional Random Fields

- Tensorflow 구현

$$p(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y}\in Y_X} e^{s(X,\tilde{y})}}$$

$$\boxed{\log p(y|X) = s(X,y) - \log \sum_{\tilde{y}\in Y_X} s(X,\tilde{y})}$$

Global normalize
= 분모에 가능한 모든 경우의
시퀀스 고려

```
def crf_log_likelihood(inputs,
                       tag_indices,
                       sequence_lengths,
                       transition_params=None):
  # Get shape information.
  num_tags = inputs.get_shape()[2].value

  # Get the transition matrix if not provided.
  if transition_params is None:
    transition_params = vs.get_variable("transitions", [num_tags, num_tags])

  sequence_scores = crf_sequence_score(inputs, tag_indices, sequence_lengths,
                                       transition_params)
  log_norm = crf_log_norm(inputs, sequence_lengths, transition_params)

  # Normalize the scores to get the log-likelihood.
  log_likelihood = sequence_scores - log_norm
  return log_likelihood, transition_params
```

Lample, G., Ballesteros, M., Subramanian, S., Kawakami, K., & Dyer, C. (2016). Neural architectures for named entity recognition. *arXiv preprint arXiv:1603.01360*.

Conditional Random Fields

- Tensorflow 구현

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}$$

Binary scores
= y_i에서 y_{i+1}로 전이할 점수

Unary scores
= i번째 레이블이 y_i일 점수

```
def crf_sequence_score(inputs, tag_indices, sequence_lengths,
                       transition_params):
  # Compute the scores of the given tag sequence.
  unary_scores = crf_unary_score(tag_indices, sequence_lengths, inputs)
  binary_scores = crf_binary_score(tag_indices, sequence_lengths,
                                   transition_params)
  sequence_scores = unary_scores + binary_scores
  return sequence_scores
```

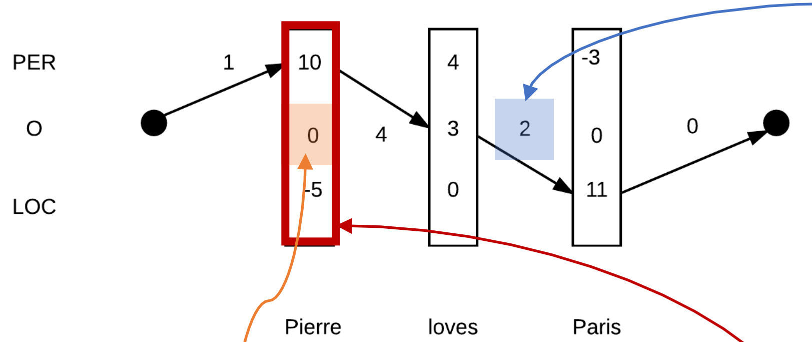
Conditional Random Fields

- $s(X, y)$

$$s(X,y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i}$$

Binary scores Unary Scores

$$1 + 10 + 4 + 3 + 2 + 11 + 0 = 31$$



직전 단어가 태그 ID=2이고
현재 단어가 태그 ID=3일 때
Binary score

문장의 1번째 단어가
태그 ID=2인 Unary score

문장의 1번째 단어에 대한
각 태그별 Unary scores

Conditional Random Fields

- P = Unary scores

예시를 위한 가정
NER 문제, 태그 종류는 3가지
Max_sequence_length(배치 데이터의 최대 토큰 길이) = 4
Seqeunce_length(현재 데이터의 토큰 길이)=3

$$P = W^T z + b = \begin{array}{cc} & \text{\# of tags} \\ \text{max seq len} & \begin{bmatrix} 5 & 1 & -1 \\ 1 & 3 & 1 \\ -1 & 4 & -1 \\ 0 & 0 & 0 \end{bmatrix} \\ & \text{padding} \end{array}$$

BiLSTM 레이어를 통과한 벡터

문장의 1번째 단어에 대한 각 태그별 Unary score

문장의 1번째 단어가 태그 ID=2인 Unary score

$Y = [1,2,2,0]$ 정답 품사 시퀀스

$$\sum_{i=1}^{n} P_{i,y_i} = P_{1,y_1=1} + P_{2,y_2=2} + P_{3,y_3=2} + P_{4,y_4=0}$$

$$= 5 + 3 + 4 + 0 = 12$$

Conditional Random Fields

- A = Binary scores

예시를 위한 가정
NER 문제, 태그 종류는 3가지
Max_sequence_length(배치 데이터의 최대 토큰 길이) = 4
Seqeunce_length(현재 데이터의 토큰 길이)=3

$$A = \begin{array}{cc} & \text{prev tag} \\ \text{curr tag} & \begin{bmatrix} -3 & 5 & -2 \\ 3 & 4 & 1 \\ 1 & \boxed{2} & 3 \end{bmatrix} \end{array}$$

# of tags 크기의 정방행렬을 랜덤 초기화 후 학습
직전 단어가 태그 ID=2이고 현재 단어가 태그 ID=3일 때 Bianry score

$Y = [1,2,2,\boxed{0}]$ 정답 품사 시퀀스

$$\sum_{i=0}^{n} A_{y_i, y_{i+1}} = A_{y_0, y_1=1} + A_{y_1=1, y_2=2} + A_{y_2=2, y_3=2} + A_{y_3=2, y_4}$$

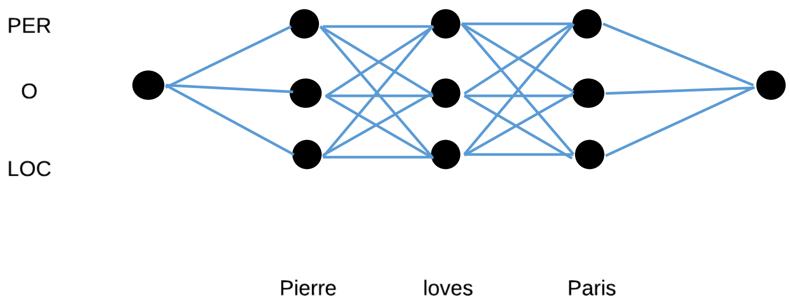
0으로 가정

$$= 3 + 4 = 7$$

Conditional Random Fields

- Global Normalize

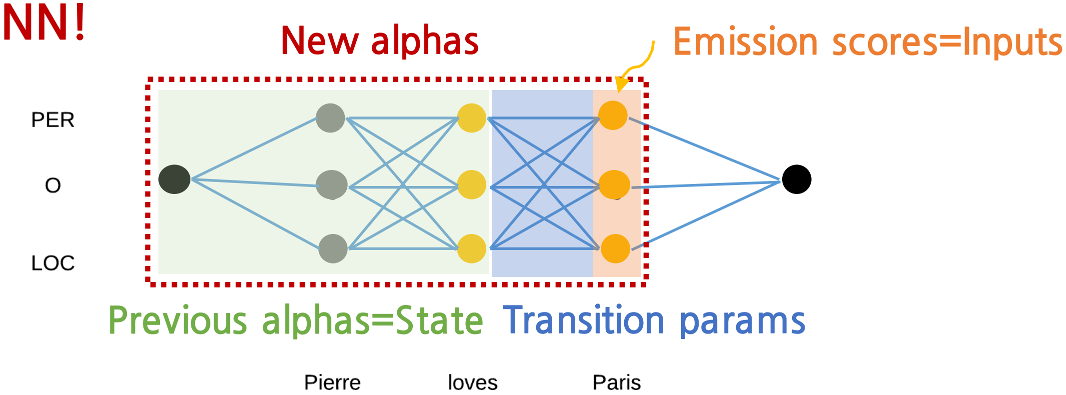
➔ 모든 경우의 수(27가지)에 대해 sequence score $s(X, y)$를 구한다



Conditional Random Fields

- Global Normalize

  ➔ 텐서플로우 구현? **RNN!**




```python
def crf_log_norm(inputs, sequence_lengths, transition_params):
  first_input = array_ops.slice(inputs, [0, 0, 0], [-1, 1, -1])
  first_input = array_ops.squeeze(first_input, [1])

  def _single_seq_fn():
    return math_ops.reduce_logsumexp(first_input, [1])

  def _multi_seq_fn():
    rest_of_input = array_ops.slice(inputs, [0, 1, 0], [-1, -1, -1])
    forward_cell = CrfForwardRnnCell(transition_params)
    _, alphas = rnn.dynamic_rnn(
        cell=forward_cell,
        inputs=rest_of_input,
        sequence_length=sequence_lengths - 1,
        initial_state=first_input,
        dtype=dtypes.float32)
    log_norm = math_ops.reduce_logsumexp(alphas, [1])
    return log_norm

  max_seq_len = array_ops.shape(inputs)[1]
  return control_flow_ops.cond(pred=math_ops.equal(max_seq_len, 1),
                               true_fn=_single_seq_fn,
                               false_fn=_multi_seq_fn)
```

```python
class CrfForwardRnnCell(rnn_cell.RNNCell):

  def __init__(self, transition_params):
    self._transition_params = array_ops.expand_dims(transition_params, 0)
    self._num_tags = transition_params.get_shape()[0].value

  @property
  def state_size(self):
    return self._num_tags

  @property
  def output_size(self):
    return self._num_tags

  def __call__(self, inputs, state, scope=None):
    state = array_ops.expand_dims(state, 2)
    transition_scores = state + self._transition_params
    new_alphas = inputs + math_ops.reduce_logsumexp(transition_scores, [1])
    return new_alphas, new_alphas
```

Conditional Random Fields

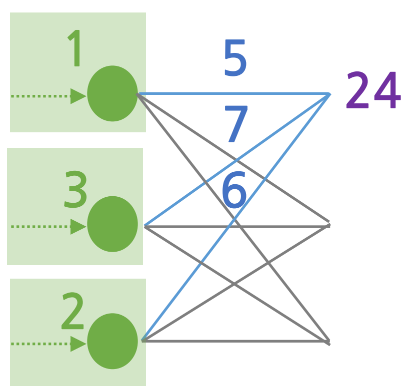
- Global Normalize

  ➔ 텐서플로우 구현? **RNN!**

$$\begin{bmatrix}1 & 3 & 2\\ 1 & 3 & 2\\ 1 & 3 & 2\end{bmatrix} + \begin{bmatrix}5 & 7 & 6\\ \circ & \circ & \circ\\ \circ & \circ & \circ\end{bmatrix} = \begin{bmatrix}6 & 10 & 8\\ \circ & \circ & \circ\\ \circ & \circ & \circ\end{bmatrix} \rightarrow [24 \quad \circ \quad \circ]$$

Expand_dim된
Previous alphas=state

Transition Params
=Binary Scores




```
class CrfForwardRnnCell(rnn_cell.RNNCell):

  def __init__(self, transition_params):
    self._transition_params = array_ops.expand_dims(transition_params, 0)
    self._num_tags = transition_params.get_shape()[0].value

  @property
  def state_size(self):
    return self._num_tags

  @property
  def output_size(self):
    return self._num_tags

  def __call__(self, inputs, state, scope=None):
    state = array_ops.expand_dims(state, 2)
    transition_scores = state + self._transition_params
    new_alphas = inputs + math_ops.reduce_logsumexp(transition_scores, [1])
    return new_alphas, new_alphas
```

Conditional Random Fields

- 학습 : 아래 log likelihood 최대화

$$\log p(y|X) = s(X, y) - \log \sum_{\tilde{y} \in Y_X} s(X, \tilde{y})$$

➔ 정답 시퀀스 y의 sequence score $s(X, y)$를 높이고
나머지 경우에 해당하는 sequence score를 낮춘다

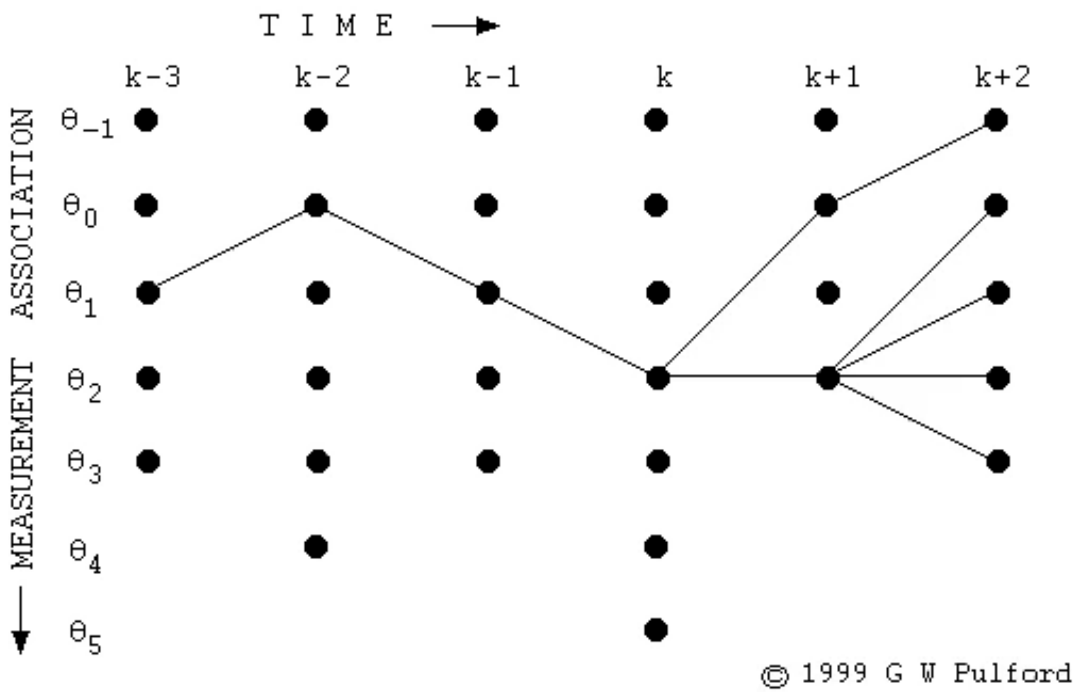
- 텐서플로우 구현

```
# scores : Bi-LSTM forward/backward output를 concat한 뒤 affine transformation
log_likelihood, transition_params = tf.contrib.crf.crf_log_likelihood(scores, labels, sequence_lengths)
loss = tf.reduce_mean(-log_likelihood)
```

Viterbi algorithm

- 디코딩 전략 (inference에서만 적용)
- Forward computation

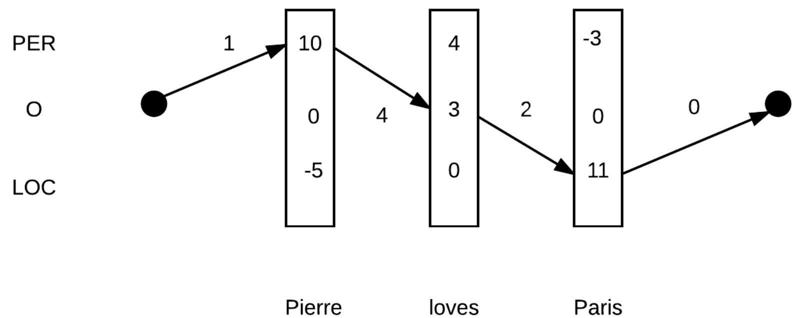
각 time step, state별로 최대 소코어를 내는 path를 모두 찾아놓는다



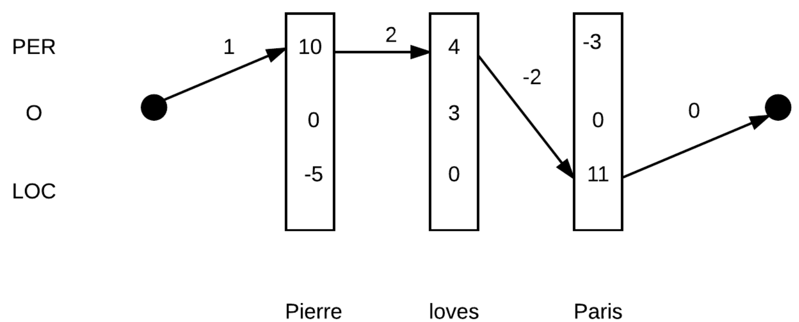
Viterbi algorithm

- Backtrace

  미리 구해놓은 path들 가운데 가장 높은 시퀀스를 택한다



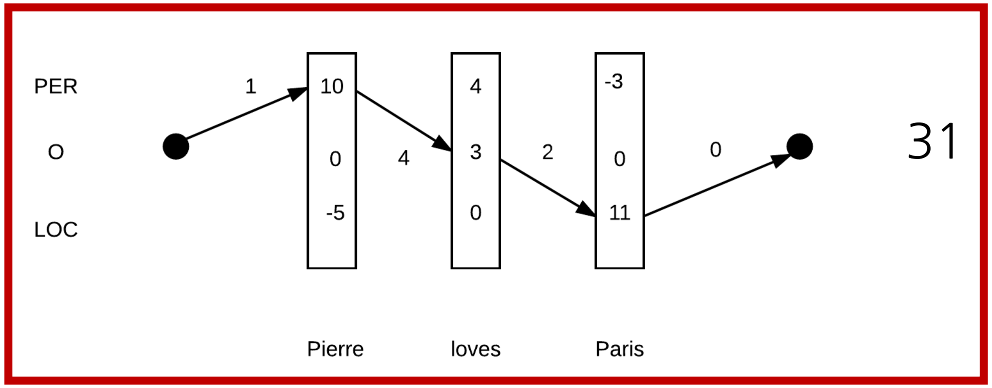
31

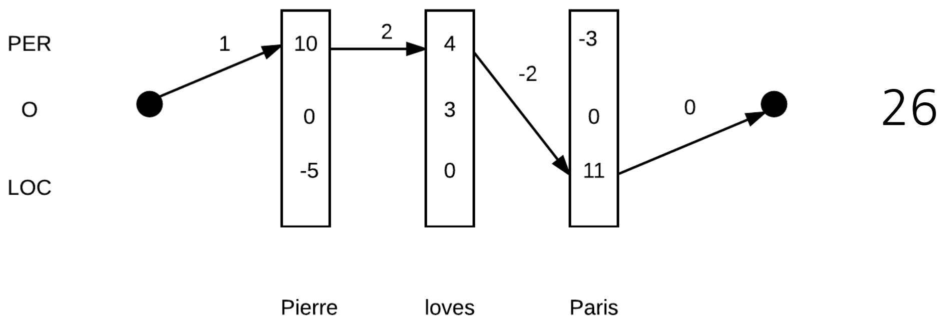


26

Viterbi algorithm

- Backtrace





Viterbi algorithm

- Tensorflow (inference용 numpy)

```python
def viterbi_decode(score, transition_params):
    trellis = np.zeros_like(score)
    backpointers = np.zeros_like(score, dtype=np.int32)
    trellis[0] = score[0]

    for t in range(1, score.shape[0]):
        v = np.expand_dims(trellis[t - 1], 1) + transition_params
        trellis[t] = score[t] + np.max(v, 0)
        backpointers[t] = np.argmax(v, 0)

    viterbi = [np.argmax(trellis[-1])]
    for bp in reversed(backpointers[1:]):
        viterbi.append(bp[viterbi[-1]])
    viterbi.reverse()

    viterbi_score = np.max(trellis[-1])
    return viterbi, viterbi_score
```

Experiments

- Word Embedding

(1) GloVe : 위키피디아, 100차원

(2) Senna : 위키피디아 + 로이터 말뭉치, 50차원

(3) Word2Vec : 구글 뉴스, 300차원

(4) Random : uniformly random $\left[-\sqrt{3/dim}, \sqrt{3/dim}\right]$

100차원

- Character Embedding

uniformly random $\left[-\sqrt{3/dim}, \sqrt{3/dim}\right]$, 30차원

Experiments

- Hyper-parameters

Layer	Hyper-parameter	POS	NER
CNN	window size	3	3
	number of filters	30	30
LSTM	state size	200	200
	initial state	0.0	0.0
	peepholes	no	no
Dropout	dropout rate	0.5	0.5
	batch size	10	10
	initial learning rate	0.01	0.015
	decay rate	0.05	0.05
	gradient clipping	5.0	5.0

Table 1: Hyper-parameters for all experiments.

- Early Stopping

50에폭 언저리에서 valid 셋 점수가 베스트일 경우

Experiments

- Data Sets (POS / NER)

Dataset		WSJ	CoNLL2003
Train	SENT	38,219	14,987
	TOKEN	912,344	204,567
Dev	SENT	5,527	3,466
	TOKEN	131,768	51,578
Test	SENT	5,462	3,684
	TOKEN	129,654	46,666

Table 2: Corpora statistics. SENT and TOKEN refer to the number of sentences and tokens in each data set.

Experiments

- Main Results : 당시 SOTA

Model	Acc.
Giménez and Màrquez (2004)	97.16
Toutanova et al. (2003)	97.27
Manning (2011)	97.28
Collobert et al. (2011)‡	97.29
Santos and Zadrozny (2014)‡	97.32
Shen et al. (2007)	97.33
Sun (2014)	97.36
Søgaard (2011)	97.50
This paper	**97.55**

Table 4: POS tagging accuracy of our model on test data from WSJ proportion of PTB, together with top-performance systems. The neural network based models are marked with ‡.

Model	F1
Chieu and Ng (2002)	88.31
Florian et al. (2003)	88.76
Ando and Zhang (2005)	89.31
Collobert et al. (2011)‡	89.59
Huang et al. (2015)‡	90.10
Chiu and Nichols (2015)‡	90.77
Ratinov and Roth (2009)	90.80
Lin and Wu (2009)	90.90
Passos et al. (2014)	90.90
Lample et al. (2016)‡	90.94
Luo et al. (2015)	91.20
This paper	**91.21**

Table 5: NER F1 score of our model on test data set from CoNLL-2003. For the purpose of comparison, we also list F1 scores of previous top-performance systems. ‡ marks the neural models.

Experiments

- CRF 효과 : 안 쓴거보다는 약간 향상

Model	POS Dev Acc.	POS Test Acc.	NER Dev Prec.	NER Dev Recall	NER Dev F1	NER Test Prec.	NER Test Recall	NER Test F1
BRNN	96.56	96.76	92.04	89.13	90.56	87.05	83.88	85.44
BLSTM	96.88	96.93	92.31	90.85	91.57	87.77	86.23	87.00
BLSTM-CNN	97.34	97.33	92.52	93.64	93.07	88.53	90.21	89.36
BRNN-CNN-CRF	97.46	97.55	94.85	94.63	94.74	91.35	91.06	91.21

Table 3: Performance of our model on both the development and test sets of the two tasks, together with three baseline systems.

Experiments

- CRF 효과 : UNK 토큰에 강건

	POS							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	97.57	**93.75**	90.29	80.27	97.55	**93.45**	90.14	80.07
LSTM-CNN-CRF	**97.68**	93.65	**91.05**	**82.71**	**97.77**	93.16	**90.65**	**82.49**
	NER							
	Dev				Test			
	IV	OOTV	OOEV	OOBV	IV	OOTV	OOEV	OOBV
LSTM-CNN	94.83	87.28	96.55	82.90	90.07	89.45	100.00	78.44
LSTM-CNN-CRF	**96.49**	**88.63**	**97.67**	**86.91**	**92.14**	**90.73**	100.00	**80.60**

Table 9: Comparison of performance on different subsets of words (accuracy for POS and F1 for NER).

임베딩 또는
학습말뭉치에
없는 단어

- OOV에 대한 POS, ENTITY 예측 정확도가 CRF를 썼을 때 약간 상승

Experiments

- 워드 임베딩 효과 : GloVe가 좋음, POS에서는 pretrain 안해도 굿

Embedding	Dimension	POS	NER
Random	100	97.13	80.76
Senna	50	97.44	90.28
Word2Vec	300	97.40	84.91
GloVe	100	**97.55**	**91.21**

Table 6: Results with different choices of word embeddings on the two tasks (accuracy for POS tagging and F1 for NER).

- Dropout 효과 : significant improvements

	POS			**NER**		
	Train	**Dev**	**Test**	**Train**	**Dev**	**Test**
No	98.46	97.06	97.11	99.97	93.51	89.25
Yes	97.86	97.46	97.55	99.63	94.74	91.21

Table 7: Results with and without dropout on two tasks (accuracy for POS tagging and F1 for NER).