ACL 2016

Named Entity Recognition with Bidirectional LSTM-CNNs

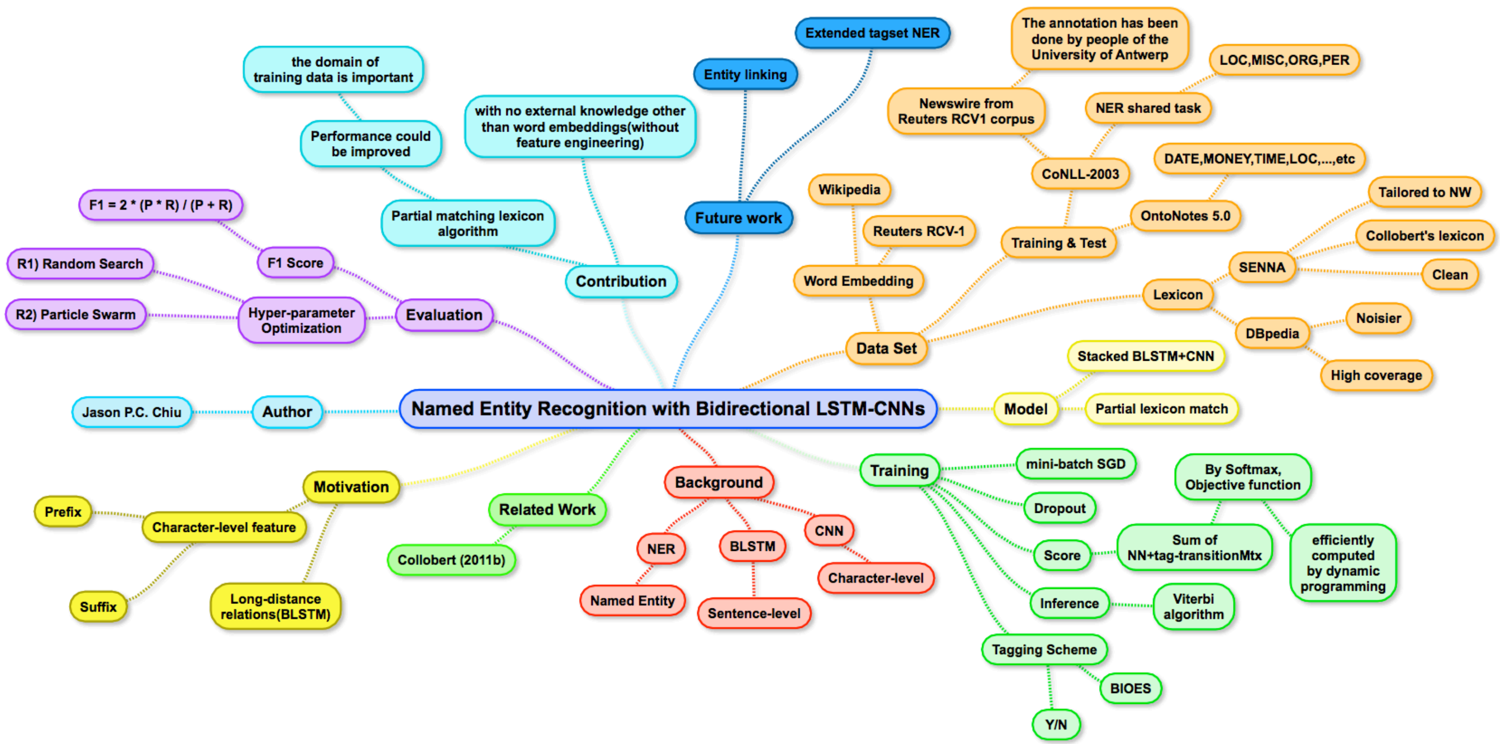
Jason P.C Chiu, and Eric Nichols

2016010662 윤주성

Contents

ACL 2016

0. Overview



1. Author

Jason Chiu

Student at The University of British Columbia
Vancouver, British Columbia, Canada | Computer Software

Previous Honda Reseach Institute Japan Co. Ltd., MS/MRI Research Group
Education The University of British Columbia / UBC

Named entity recognition with bidirectional lstm-cnns [PDF] arxiv.org
JPC Chiu, E Nichols - arXiv preprint arXiv:1511.08308, 2015 - arxiv.org
Abstract: Named entity recognition is a challenging task that has traditionally required large amounts of knowledge in the form of feature engineering and lexicons to achieve high performance. In this paper, we present a novel neural network architecture that ...
11회 인용 관련 학술자료 전체 3개의 버전 인용 저장

[PDF] Sequential Labeling with Bidirectional LSTM-CNNs [PDF] anlp.jp
JPC Chiu, E Nichols - 2016 - anlp.jp
Part-of-speech tagging, text chunking, and named entity recognition are fundamental tasks in NLP. High performance approaches have been dominated by applying statistical models such as CRF, SVM, or perceptron models to hand-crafted features [24, 37, 12, 27, 23]. ...
관련 학술자료 인용 저장 더보기

2. Introduction

2.1 Motivation

기존 연구의 한계점

1) Use a simple feed-forward neural network

각 단어 주변으로 Fixed sized window를 사용함으로써
단어사이의 Long-distance relations 고려하지 못함

2) Depend solely on word embeddings

워드 임베딩에만 의존함으로써 Character level feature인
Prefix, suffix등을 이용하지 못함

2. Introduction

2.1 Motivation

기존 연구의 한계점 극복

1) BLSTM

Bidirectional LSTM을 통해 infinite amount of context를 고려하자

2) CNN

Convolutional Neural Network를 통해 character level의 Feature를 얻어내자

3.1 NER

3.1.1 Named Entity

Person, location, organizations, products 등의 Object에 특별히 할당된 특유의 이름이 있는 Entity

"Obama is the president of the United States".

Specific object - Named Entity(O)

Different objects in different worlds - Named Entity(X)

Contents

3.1 NER

3.1.2 Named Entity Recognition이란?

Information extraction의 Task중 하나

Named entity in text —classify→ **Pre-defined categories)**
person, organizations, locations, time, quantities, monetary values, percentage, etc

Example)

Jim bought 300 shares of Acme Corp. in 2006.

->[Jim]$_{Person}$ bought 300 shares of [Acme Corp.]$_{Organization}$ in [2006]$_{Time}$.

Contents

3.1 NER

3.1.2 Named Entity Recognition이란?

NER is typically viewed as a sequential prediction problem

HMM (Rabiner, 1989)	CRF (Lafferty, 2001)	Perceptron (Collins, 2002)	➡	BLSTM-CNNs (제안하는 방법)

Sequential Prediction Problem

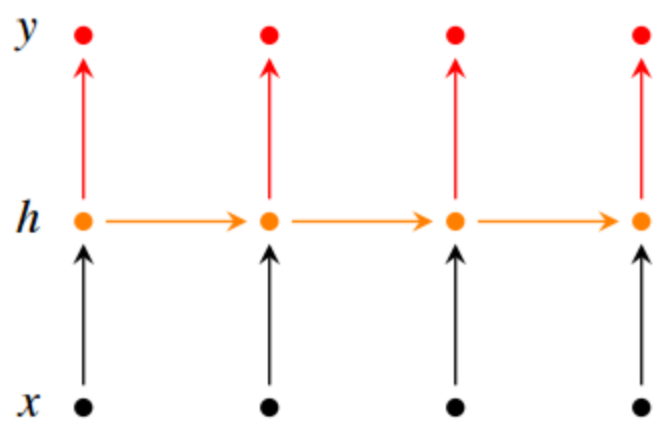
$$P([y]_1^T \mid [x]_1^T, \theta')$$

Input sequence $[x]_1^T$와 parameter θ'가 있을때 True Tag sequence $[y]_1^T$의 확률

Contents

3.2 BLSTM

LSTM



Past words로
다음단어를 예측해보자

$$h_t = f(Wx_t + Vh_{t-1} + b)$$
$$y_t = g(Uh_t + c)$$

왜 LSTM을 쓰는가?　RNN -> LSTM

RNN은 이론상
Long-term dependency를 갖지만
실제적으로는 잘 안됨

이를 보완하기 위해서

(More complex units for activation)

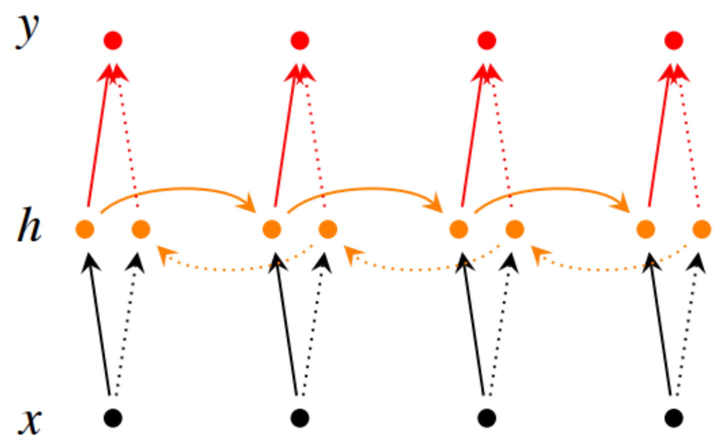
을 사용한 것이 LSTM

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1}) \quad \text{(Input gate)}$$
$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1}) \quad \text{(Forget gate)}$$
$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1}) \quad \text{(Output/Exposure gate)}$$
$$\tilde{c}_t = \tanh(W^{(c)}x_t + U^{(c)}h_{t-1}) \quad \text{(New memory cell)}$$
$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \quad \text{(Final memory cell)}$$
$$h_t = o_t \circ \tanh(c_t)$$

Contents

3.2 BLSTM

BLSTM



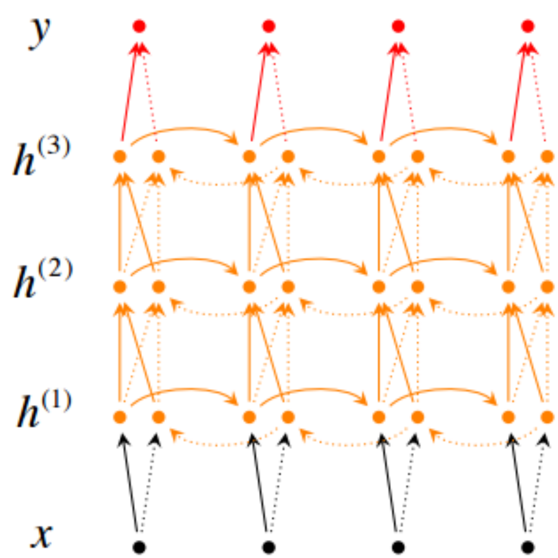
Future words로도
단어를 예측해보자

$$\overrightarrow{h_t} = f(\overrightarrow{W}x_t + \overrightarrow{V}\overrightarrow{h}_{t-1} + \overrightarrow{b})$$

$$\overleftarrow{h_t} = f(\overleftarrow{W}x_t + \overleftarrow{V}\overleftarrow{h}_{t+1} + \overleftarrow{b})$$

$$y_t = g(U[\overrightarrow{h_t};\overleftarrow{h_t}] + c)$$

Stacked BLSTM

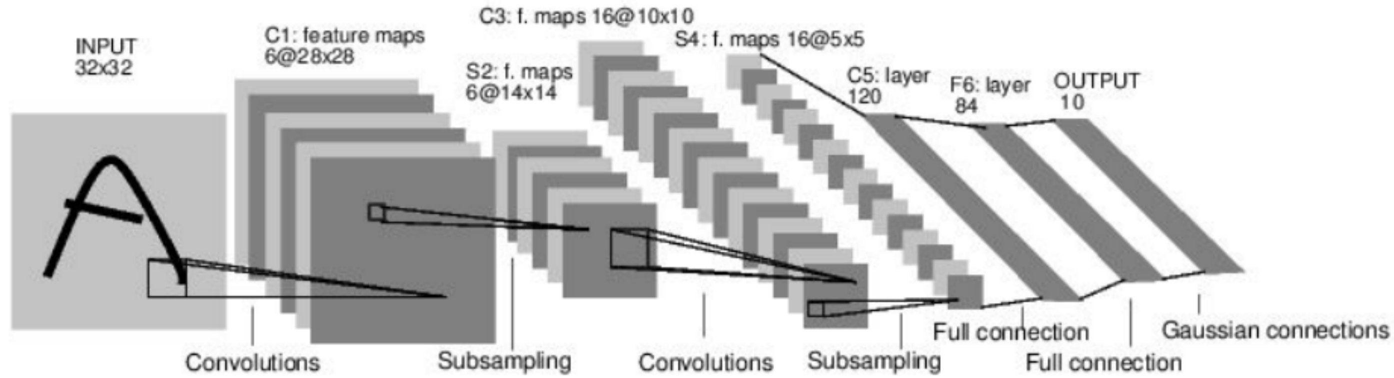


BLSTM의 Hidden Layer층을
더 추가해서 Neural Network의
성능을 높여보자

Stacked BSLTM
본 논문에서 사용

Contents

3.3 CNN



[LeNet-5, LeCun 1980]

Neural Network의 한 종류, receptive field를 이용하여 Input data의 Local Feature를 효과적으로 얻어냄

Input Layer, Convolutional Layer, Pooling Layer, Fully Connected Layer, Output Layer로 구성됨

영상, 음성 분야에서 뛰어난 성능을 보이며 NLP 분야에서도 Embedding Matrix의 통해 많이 사용됨

Contents

3.3 CNN

3.3.1 Convolutional Layer

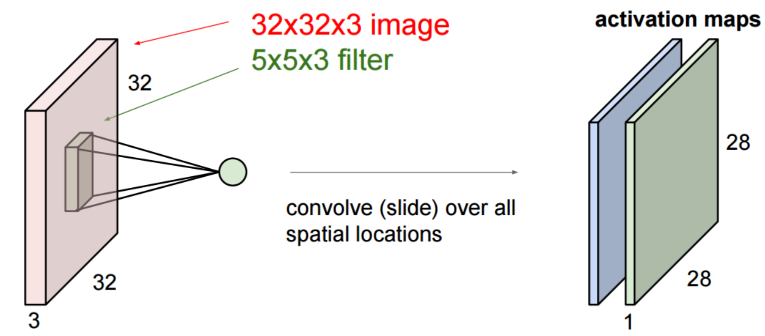
Softmax

Fully Connected Layer

Pooling Layer

Activation function
ex)ReLU, tanh

Convolutional Layer



-N개의 Filter(Receptive Field)를 사용해서 N개의 Activation map을 만듬
-Filter는 Weight sharing을 통해 Parameter 수를 줄인 구조, Overfitting을 막을 수 있음
-Filter는 학습을 통해 특정 패턴을 잡아낼 수 있음

Contents

3.3 CNN

3.3.1 Convolutional Layer

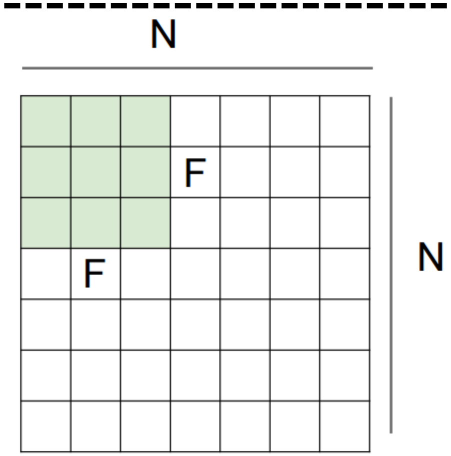
Softmax

Fully Connected Layer

Pooling Layer

Activation function
ex)ReLU, tanh

Convolutional Layer



Activation map의 size:
(N - F) / stride + 1

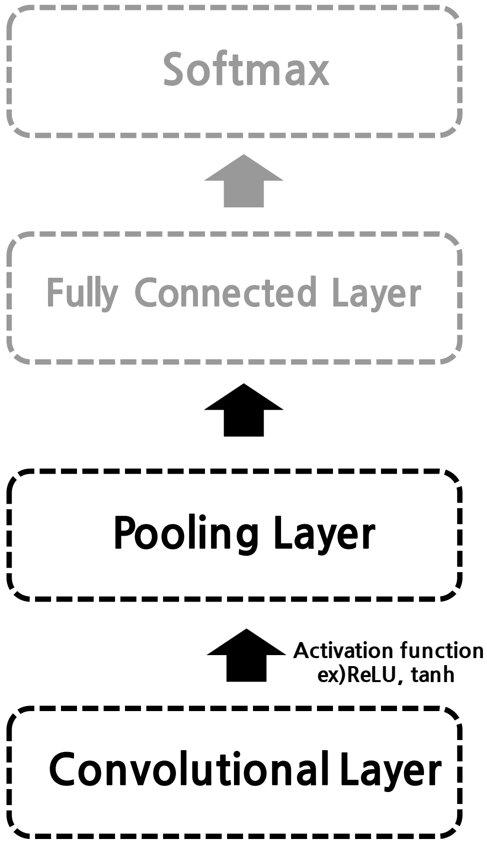
N: Padding을 포함 Input data의 크기
F: Filter Size
Stride: Filter가 이동하는 단위
Ex) N = 7, F = 3
Stride 1 : (7 - 3) / 1 + 1 = 5
Stride 2 : (7 - 3) / 2 + 1 = 3

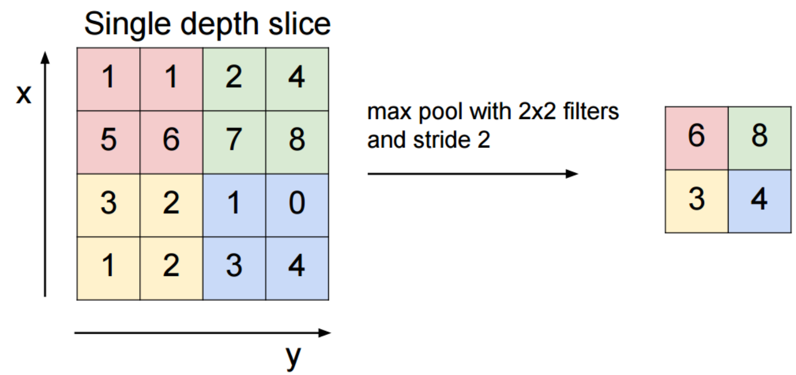
Activation map의 크기는 Filter의 크기 및 Stride(건너뛰기) 등
Hyper parameter에 의해 달라짐

Contents

3.3 CNN

3.3.2 Pooling Layer





- Down-sampling을 통해
  Spatial size of representation을 줄임
- Parameter 개수 및 Computation을 줄여줌
- Overfitting을 막아줌

Contents

3.3 CNN

3.3.3 Fully Connected Layer

Softmax

↑

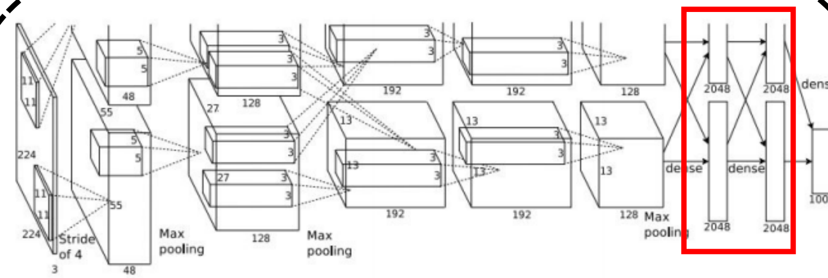
Fully Connected Layer

↑

Pooling Layer

↑ Activation function ex)ReLU, tanh

Convolutional Layer



- Convolutional Layer와 Pooling Layer를 지나면서 얻어진 High level feature가 Fully connected Neural Network의 Input으로 들어감
- High level feature data를 구분해주는 역할

Contents

3.3 CNN

3.3.4 Softmax

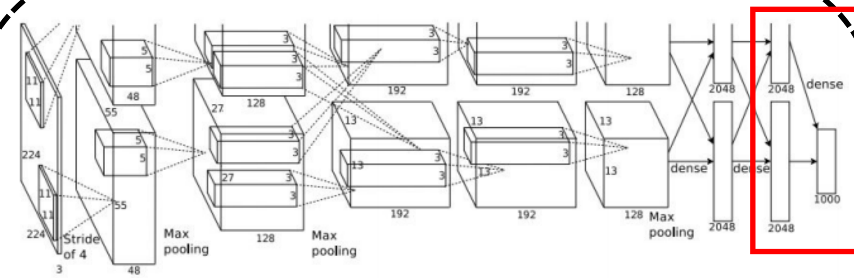
Softmax

Fully Connected Layer

Pooling Layer

Activation function
ex)ReLU, tanh

Convolutional Layer



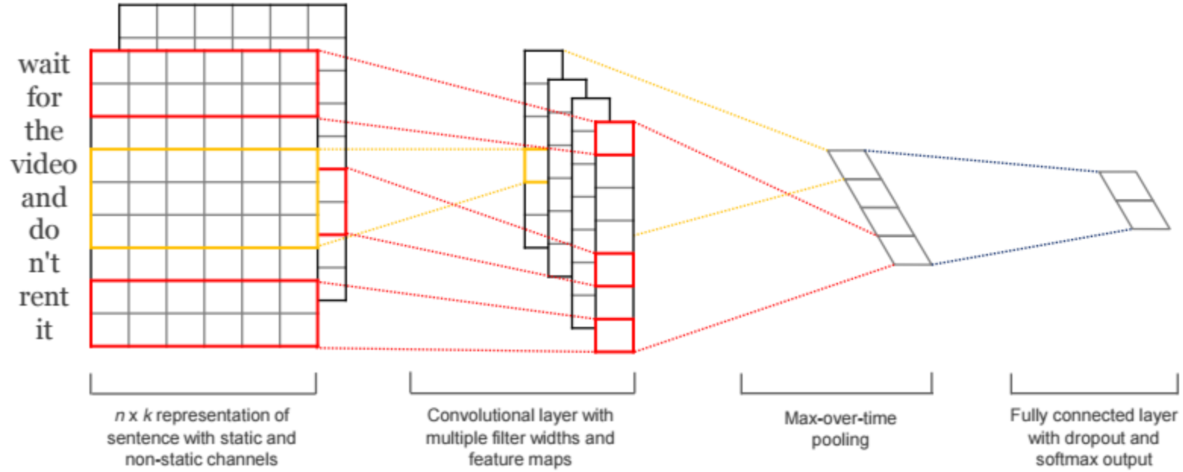
- Softmax를 통해 최종적으로 Classification

Contents

3.3 CNN

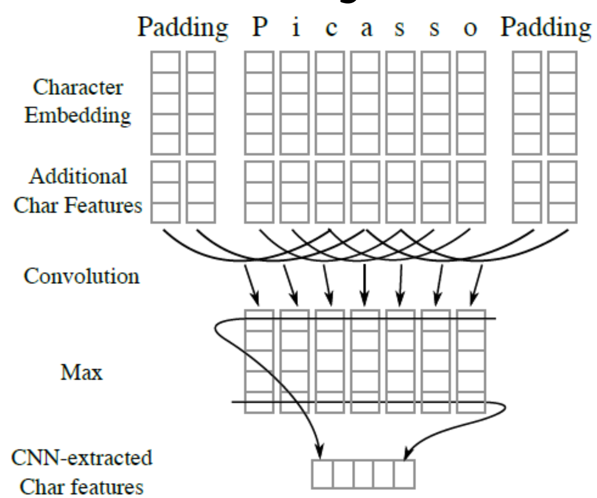
3.3.5 NLP에서 CNN

Word embedding



[Figure from Kim(2014)]

Character embedding



Input data

- Embedding된 Vector

Contents

4.1 Dataset

Use to train word embedding

Wikipedia (from 2007, 2011)

Reuters RCV1 datasets
(https://archive.ics.uci.edu/ml/datasets/Reuters+RCV1+RCV2+Multilingual,+Multiview+Text+Categorization+Test+collection#)

Use to add word-level feature (Lexicon)

DBpedia (http://wiki.dbpedia.org/Datasets)
Use to build lexicon dataset in this paper (Noisier but high coverage)

SENNA system (http://ronan.collobert.com/senna/)
Lexicon dataset that used in Collobert 2011b (Clean & Tailored to NW)

Use to train and test model

CoNLL-2003 dataset (http://www.cnts.ua.ac.be/)
LOC,MISC,ORG,PER의 Tag로 구성, 사람이 직접 정리한 정돈된 데이터

OntoNotes 5.0 tagset (https://catalog.ldc.upenn.edu/LDC2013T19)
DATE, MONEY, TIME, LOC, GPE, ORG, LANG, LAW 등 다양한 Tag로 구성

※ SENNA (Semantic/syntactic Extraction using a Neural Network

4.1 Dataset

Use to train word embedding

Wikipedia (from 2007, 2011)

Reuters RCV1 datasets
(https://a ... 2+Multi
lingual,+

Dataset	Train	Dev	Test
CoNLL-2003	204,567 (23,499)	51,578 (5,942)	46,666 (5,648)
OntoNotes 5.0 / CoNLL-2012	1,088,503 (81,828)	147,724 (11,066)	152,728 (11,257)

Table 2: Dataset sizes in number of tokens (entities)

feature
Lexicon)

DBpedia
Use ... rage)

SENNA system(http://ronan.collobert.com/senna/)
Lexicon dataset that used in Collobert 2011b (Clean & Tailored to NW)

Use to train and test model

CoNLL-2003 dataset (http://www.cnts.ua.ac.be/)
LOC,MISC,ORG,PER의 Tag로 구성, 사람이 직접 정리한 정돈된 데이터

OntoNotes 5.0 tagset (https://catalog.ldc.upenn.edu/LDC2013T19)
DATE, MONEY, TIME, LOC, GPE, ORG, LANG, LAW 등 다양한 Tag로 구성

※ SENNA (Semantic/syntactic Extraction using a Neural Network

Contents

4.2 Dataset example

단어	POS tag	Syntactic chunk tag	NER tag
U.N.	NNP	I-NP	I-ORG
official	NN	I-NP	O
Ekeus	NNP	I-NP	I-PER
heads	VBZ	I-VP	O
for	IN	I-PP	O
Baghdad	NNP	I-NP	I-LOC
.	.	O	O

Example of CoNLL-2003 Dataset

4가지 Name Entities Tag로 구성
(LOC,MISC,ORG,PER)

Training / Dev
News feed
From *August* 1996

Test
News feed
From *December* 1996

Reuters RCV1 corpus의
Newswire data

OntoNotes에 비해
데이터수가 적음

```
Treebanked sentence:
--------------------
    I ground the rye on number 6 click -LRB- out of 8 -RRB- in my Champion Juicer grinder .

Tree:
-----
    (TOP (S (NP-SBJ (PRP I))
            (VP (VBD ground)
                (NP (DT the)
                    (NN rye))
                (PP-MNR (IN on)
                        (NP (NP (NML (NN number)
                                     (CD 6))
                                (NN click))
                            (-LRB- -LRB-)
                            (PP (IN out)
                                (PP (IN of)
                                    (NP (CD 8))))
                            (-RRB- -RRB-)))
                (PP-LOC (IN in)
                        (NP (PRP$ my)
                            (NML (NNP Champion)
                                 (NNP Juicer))
                            (NN grinder))))
```

Example of OntoNotes 5.0 Dataset

DATE, MONEY, TIME, LOC, GPE,
ORG, LANG, LAW, … 다양한 Tag

Broadcast conversation, news,
magazine, web text, …

CoNLL-2003보다 다양함
데이터 수가 많음

NE tag 개수: 총 18개

Contents

4.3 Dataset Preprocessing

All Dataset

All digit sequences $\xrightarrow{\text{모두 0으로 대체}}$ Single “0”

Sentences $\xrightarrow{\text{Training 위해 토큰 길이별로 그룹}}$ Mini-batches

Word embedding

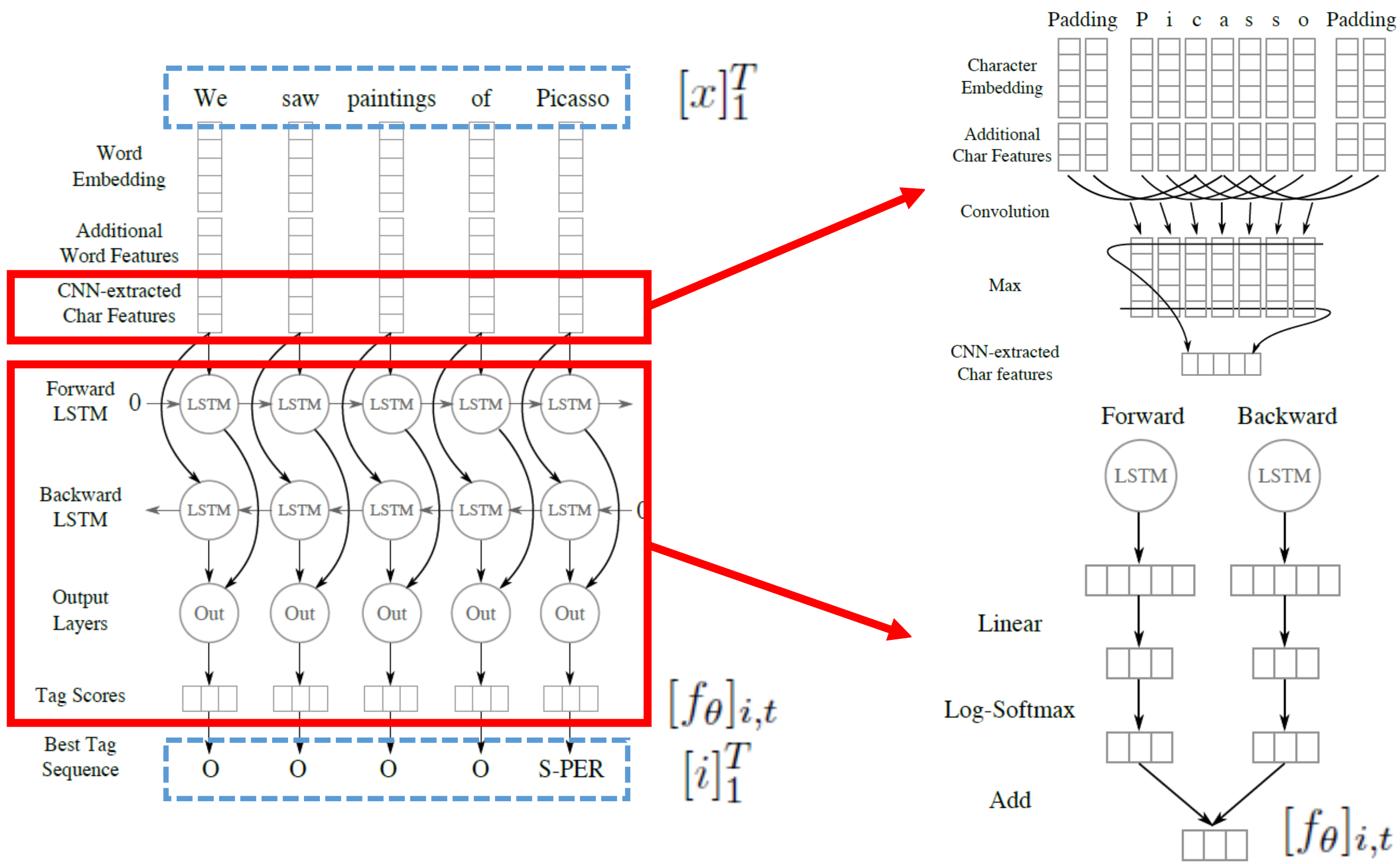
All words $\xrightarrow{\text{Lookup table에는 소문자만 사용으로 통일}}$ Lower-case

OntoNotes Dataset은 Date, Time, Money, Percentage등
다양한 Named Entity tags 때문에 digit 앞 뒤로 Split

Contents

- Author
- Introduction
- Background
- Data Set
- Model
- Training
- Result
- Conclusion

5.1 BLSTM- CNNS Model



Sentence $[x]_1^T$

Word Embedding

Additional Word Feature

CNN Character Embedding

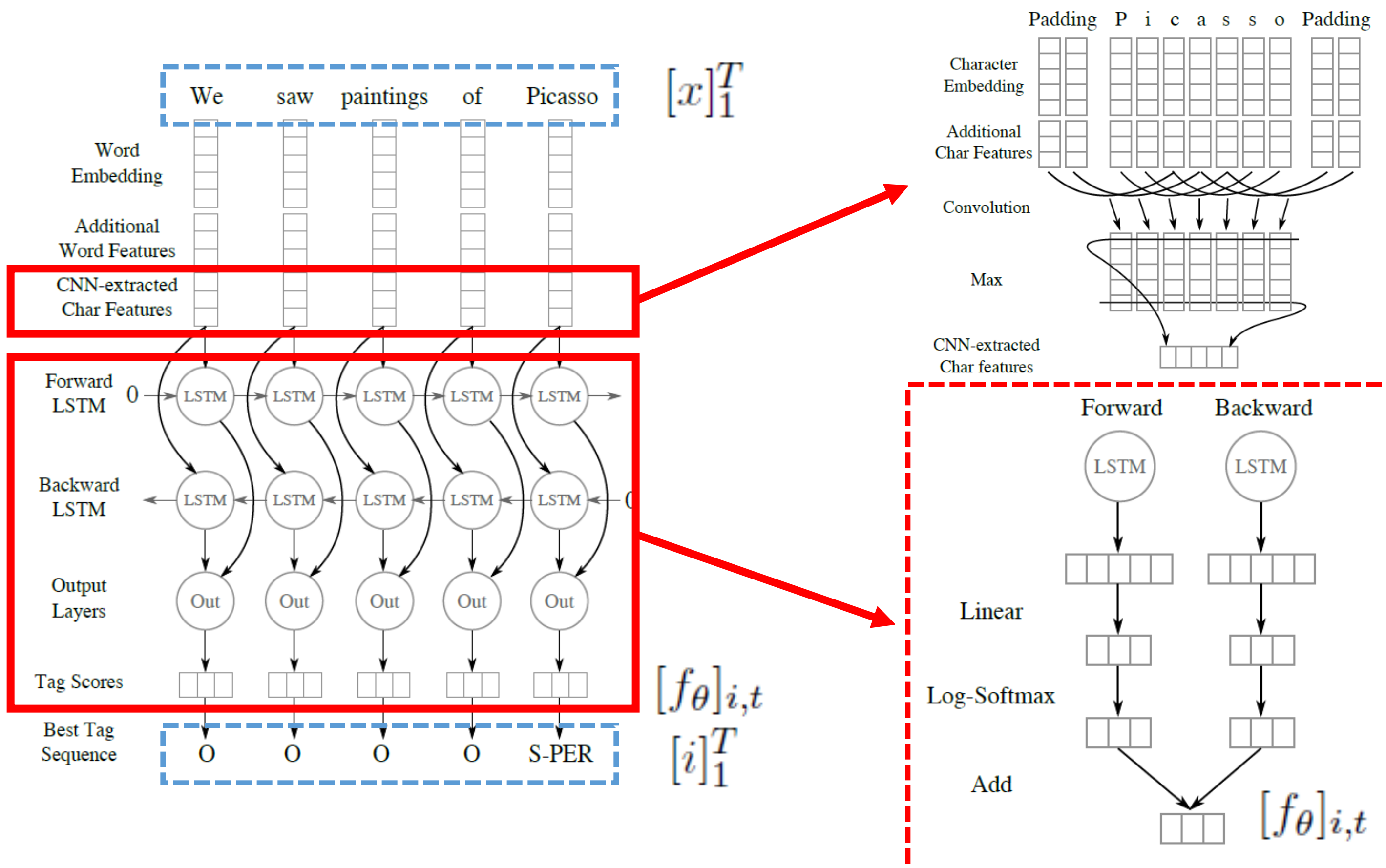
Stacked BLSTM $[f_\theta]_{i,t}$

Seq of Tag Scores $S([x]_1^T, [i]_1^T, \theta')$

Viterbi Algorithm $[i]_1^T$

CNN과 Stacked BLSTM은 동시에 Training 됨

5.1 BLSTM- CNNS Model



Sentence $[x]_1^T$

Word Embedding

Additional Word Feature

CNN Character Embedding

Stacked BLSTM $[f_\theta]_{i,t}$

Seq of Tag Scores $S([x]_1^T, [i]_1^T, \theta')$

Viterbi Algorithm $[i]_1^T$

CNN과 Stacked BLSTM은 동시에 Training 됨

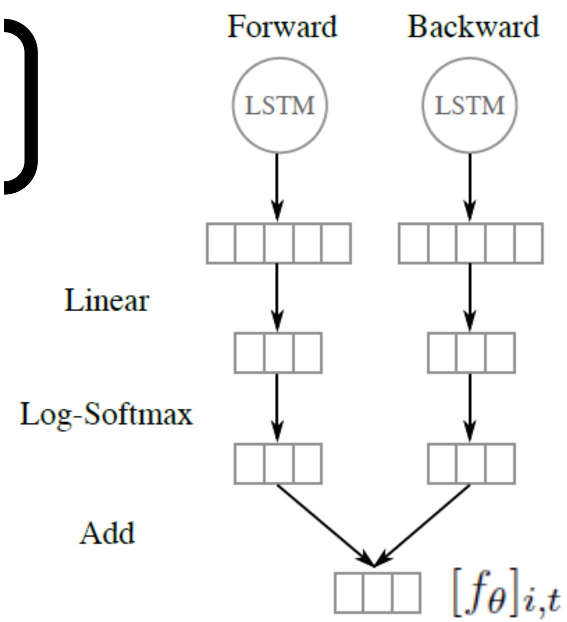
Contents

5.1 BLSTM- CNNS Model

각각 LSTM의 output 값을 각각 Linear layer와
각각 Log-Softmax layer로 구성된 Neural network의 입력값으로 사용

Neural-network의 최종 결과값?
tag category vector(tag score) $[f_\theta]_{i,t}$

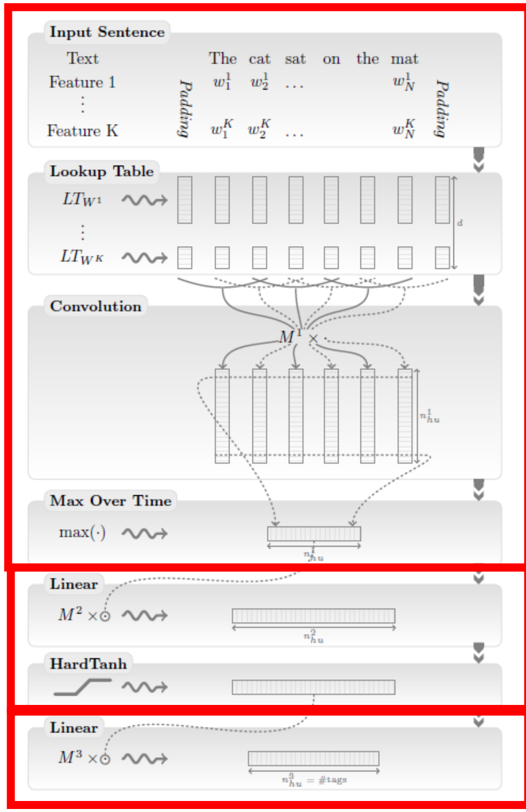
단순히, 두 vector를 더하는 것으로
최종 tag score 산출 $[f_\theta]_{i,t}$



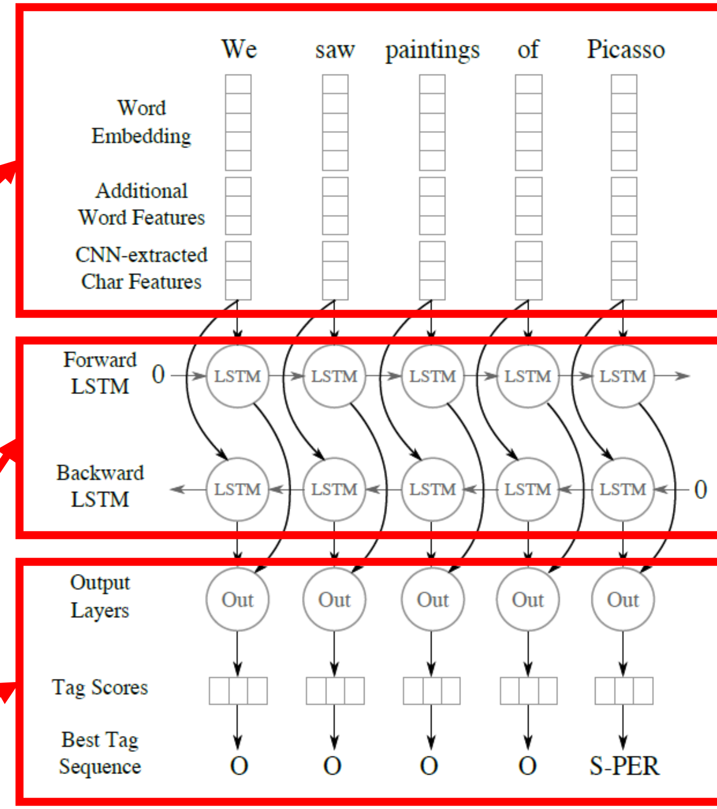
Contents

5.1 BLSTM- CNNS Model

5.1.2 Model compares between Collobert(2011b) and BLSTM-CNNs



Sentence approach network model from Collobert(2011b)



Proposed model (BLSTM-CNNs)

Collobert (2011b)

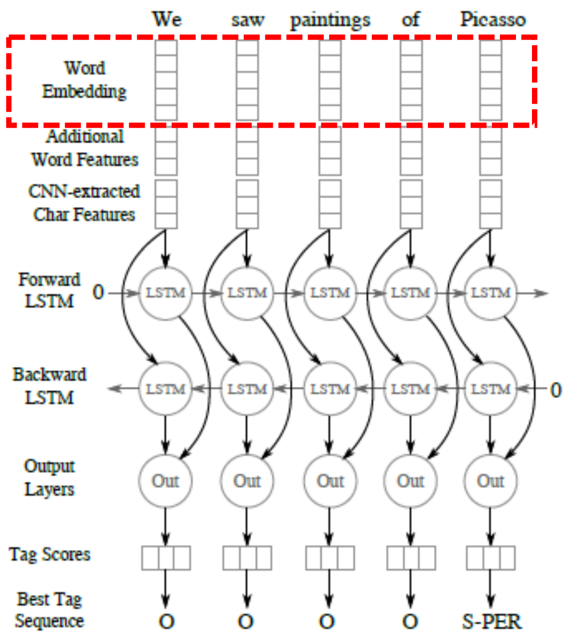
단어를 Lookup tables을 사용하여 vector로 변환
Feed-Forward Neural Network를 사용

In this paper

FFNN 대신 Stacked BLSTM 사용
CNN을 사용하여 character-level feature를 추정

5.2 Core feature

5.2.1 Word Embeddings



1) 50-dim word embeddings (by Collobert(2011b))

- Wikipedia(2007) & Reuters RCV-1 corpus.

2) Stanford's GloVe embeddings (Pennington 2014)

Original - Wikipedia & Web text

Modified - Wikipedia(2011) & Reuters RCV-1 corpus.

3) Google's word2vec embeddings (Mikilov 2013)

Original - Google news

Modified - Wikipedia(2011) & Reuters RCV-1 corpus.

3종류의 다른 embedding 모델로 실험

2,3번 임베딩 모델을 다시 Training 후 성능향상 됨

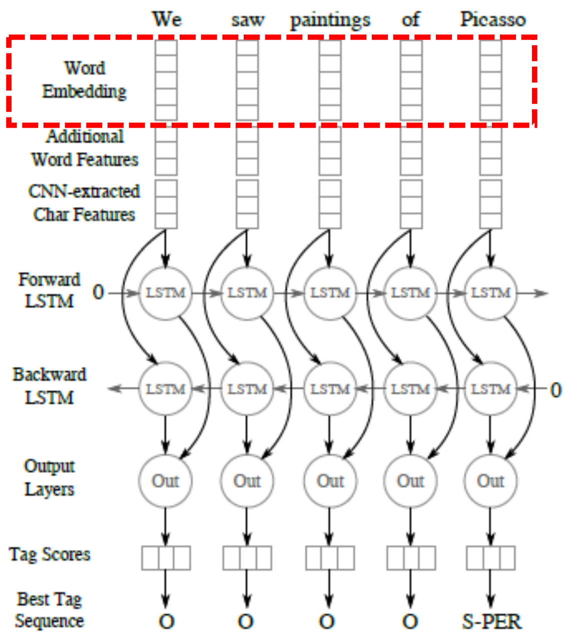
In-domain text 가 더 좋은 성능을 낸다는 가정 (Wikipedia(2011) & Reuters RCV-1 corpus)

모든 단어는 Lookup table을 통과하기 전에 대문자를 소문자로 변환

Contents

5.2 Core feature

5.2.1 Word Embeddings



1) 50-dim word embeddings (by collobert(2011b))

- Wikipedia(2007) & Reuters RCV-1 corpus.

Word Embeddings	CoNLL-2003	OntoNotes
Random 50d	87.77 (± 0.29)	83.82 (± 0.19)
Random 300d	87.84 (± 0.23)	83.76 (± 0.37)
GloVe 6B 50d	91.09 (± 0.15)	86.25 (± 0.24)
GloVe 6B 300d	90.71 (± 0.21)	86.26 (± 0.30)
Google 100B 300d	90.60 (± 0.23)	85.34 (± 0.25)
Collobert 50d	**91.62** (± 0.33)	**86.28** (± 0.26)
Our GloVe 50d	91.41 (± 0.21)	86.24 (± 0.35)
Our Skip-gram 50d	90.76 (± 0.23)	85.70 (± 0.29)

Table7. F1 scores

그러나,

Collobert의 50d 워드임베딩이 가장 좋은 성능을 나타냄

Collobert 50d 워드임베딩을 사용하기로 함

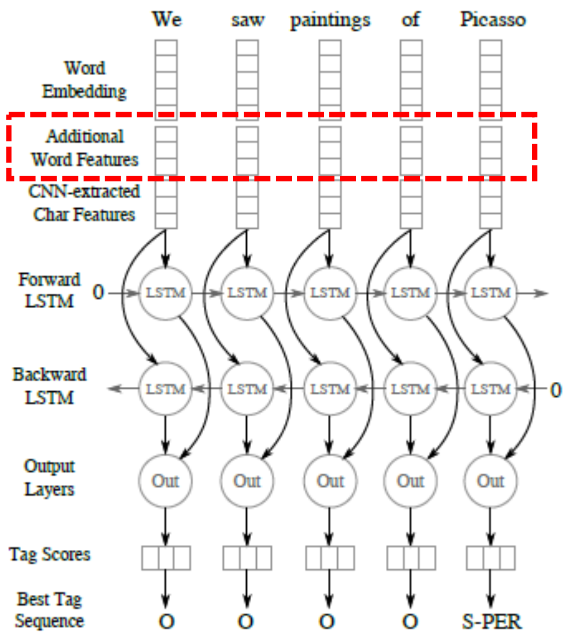
Contents

ACL 2016

5.2 Core feature

5.2.2 Additional Word-level Features

Capitalization Features



왜 쓰는 건가?

현재 논문의 Word embedding은

모든 대문자 정보를 **제거**한 상태.

이 논문에서는 5가지 옵션을 추가해서, Collobert가 제안한 방법을 **발전** 시키기 위해서

AllCaps : 모든 문자가 대문자

UpperInitial : 첫 문자만 대문자

Lowercase : 모든 문자가 소문자

Mixedcaps : 대문자와 소문자가 섞여 있음

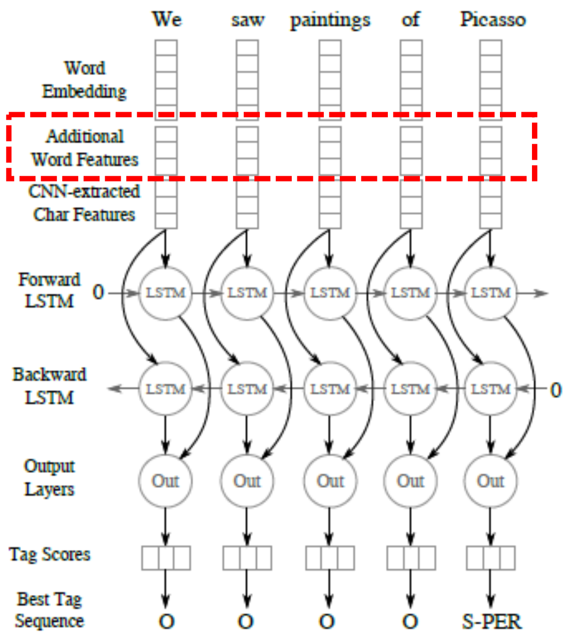
Noinfo : 정보가 없음

character-level CNN 및 character type feature와 비교하여 옵션을 검사

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons



DBpedia로부터 Named Entity List를 추출

External knowledge로 사용

카테고리는 CoNLL-2003 tags를 따름
(Person, Organization, Location, Misc)

OntoNotes tagset과 호환되는 tag는 생성하지 않음
(DBpedia와 OntoNotes 태그가 맞는게 거의 없음)

DBpedia와 SENNA Lexicon 모두 실험에 사용함

Category	SENNA	DBpedia
Location	36,697	709,772
Miscellaneous	4,722	328,575
Organization	6,440	231,868
Person	123,283	1,074,363
Total	171,142	2,344,578

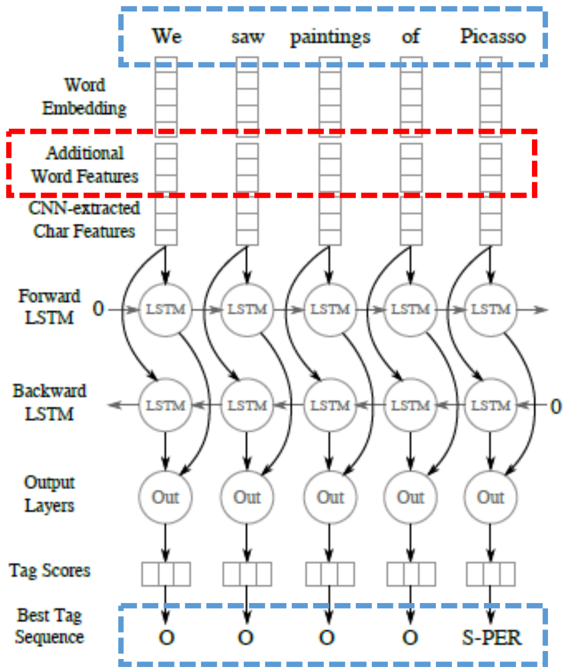
Table1. Number of entries for SENNA(collobert 2011b) & DBpedia(this paper)

Contents

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons



각각의 토큰은 BIOES(Begin, Inside, Outside, End, Single) annotation 방식으로 인코딩 됨
BIOES는 토큰의 위치를 표현

각각의 카테고리(Tag)에 대하여 토큰 생성
길이가 2토큰보다 작은 partial matches는 거짓된 정보일 가능성이 높기 때문에 무시 (단 Person은 제외)

Matching Priority

1) Exact matches > Longer partial matches > Shorter partial matches
2) 만약 두 match의 길이가 같을 경우 Earlier matches > later matches

Text	Hayao	Tada	,	commander	of	the	Japanese	North	China	Area	Army
LOC	–	–	–	–	–	B	I	–	S	–	–
MISC	–	–	–	S	B	B	I	S	S	S	S
ORG	–	–	–	–	–	B	I	B	I	I	E
PERS	B	E	–	–	–	–	–	–	S	–	–

※ '—'은 'outside' 토큰(O)을 의미, N-gram 사용

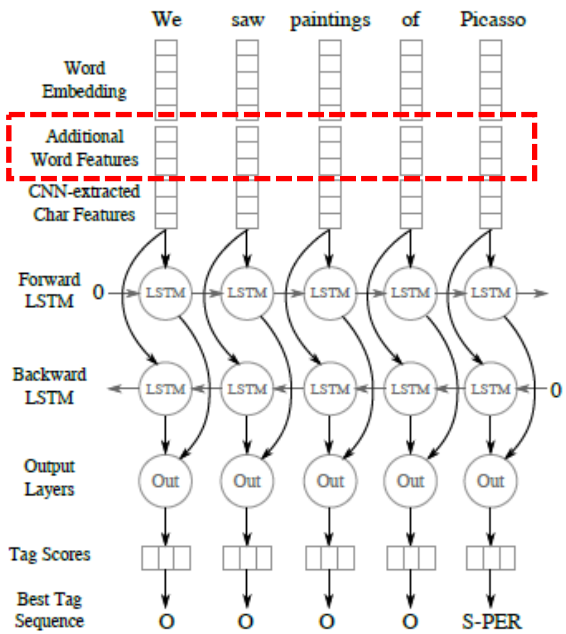
Figure 4.

Contents

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons



다양한 방법으로 Lexicon feature의 성능을 실험함

Collobert's method setting은 아래와 같음

1) Partial match와 exact match를 동일하게 취급
2) Prefix matches만 취급
3) 매우 짧은 partial match를 취급
   (길이가 2 이하라 무시했던 토큰들)
4) YES/NO(or IO) 방법을 사용 (not BIOES)

결과는?

Lexicon	Matching	Encoding	CoNLL-2003	OntoNotes
No lexicon	-	-	83.38 (± 0.20)	82.53 (± 0.40)
SENNA	Exact	YN	86.21 (± 0.39)	83.24 (± 0.33)
	Exact	BIOES	86.14 (± 0.48)	83.01 (± 0.52)
DBpedia	Exact	YN	84.93 (± 0.30)	83.15 (± 0.26)
	Exact	BIOES	85.02 (± 0.23)	83.39 (± 0.39)
	Partial	YN	85.72 (± 0.45)	83.25 (± 0.33)
	Partial	BIOES	86.18 (± 0.56)	**83.97** (± 0.38)
	Collobert's method		85.01 (± 0.31)	83.24 (± 0.26)
Both	Best combination		**87.77** (± 0.29)	83.82 (± 0.19)

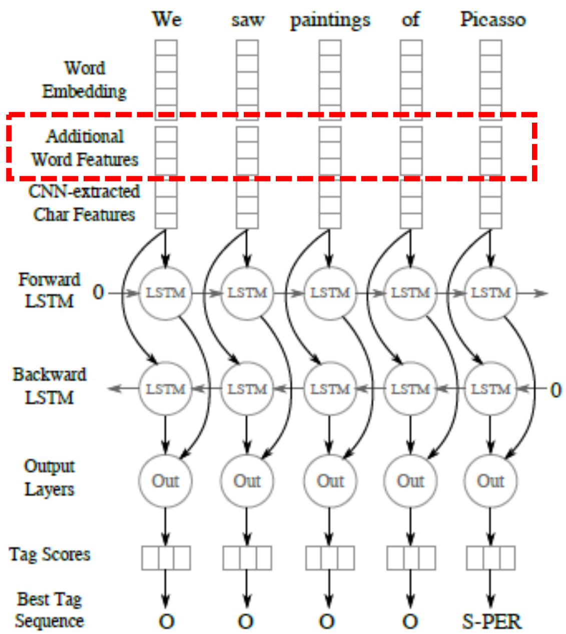
Table 9: Comparison of lexicon and matching/encoding methods over the BLSTM-CNN model employing random embeddings and no other features. When using both lexicons, the best combination of matching and encoding is Exact-BIOES for SENNA and Partial-BIOES for DBpedia. Note that the SENNA lexicon already contains "partial entries" so exact matching in that case is really just a more primitive form of partial matching.

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons

1. Matching & Encoding 방법관점



Best combination model

If(Exact matching) then SENNA lexicon
If(Partial matching) then DBpedia lexicon

Encoding은 BIOES annotation 사용

두 Lexicon을 보완해서 쓰는 것이 좋다

Lexicon	Matching	Encoding	CoNLL-2003	OntoNotes
No lexicon	-	-	83.38 (± 0.20)	82.53 (± 0.40)
SENNA	Exact	YN	86.21 (± 0.39)	83.24 (± 0.33)
	Exact	BIOES	86.14 (± 0.48)	83.01 (± 0.52)
DBpedia	Exact	YN	84.93 (± 0.30)	83.15 (± 0.26)
	Exact	BIOES	85.02 (± 0.23)	83.39 (± 0.39)
	Partial	YN	85.72 (± 0.45)	83.25 (± 0.33)
	Partial	BIOES	86.18 (± 0.56)	**83.97** (± 0.38)
	Collobert's method		85.01 (± 0.31)	83.24 (± 0.26)
Both	Best combination		**87.77** (± 0.29)	83.82 (± 0.19)

Table 9: Comparison of lexicon and matching/encoding methods over the BLSTM-CNN model employing random embeddings and no other features. When using both lexicons, the best combination of matching and encoding is Exact-BIOES for SENNA and Partial-BIOES for DBpedia. Note that the SENNA lexicon already contains "partial entries" so exact matching in that case is really just a more primitive form of partial matching.

Contents

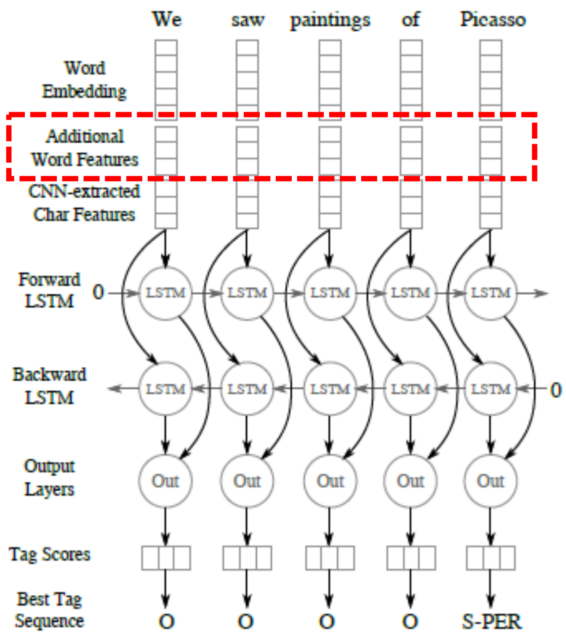
ACL 2016

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons

2. Data와 Lexicon feature 관점



Lexicon features

SENNA -> CoNLL에서 좋은 성능

Clean / Tailored to NW

DBpedia -> OntoNotes에서 좋은 성능

Noisier / Broader coverage

왜 그런가?

Lexicon	Matching	Encoding	CoNLL-2003	OntoNotes
No lexicon	-	-	83.38 (± 0.20)	82.53 (± 0.40)
SENNA	Exact	YN	86.21 (± 0.39)	83.24 (± 0.33)
	Exact	BIOES	86.14 (± 0.48)	83.01 (± 0.52)
DBpedia	Exact	YN	84.93 (± 0.30)	83.15 (± 0.26)
	Exact	BIOES	85.02 (± 0.23)	83.39 (± 0.39)
	Partial	YN	85.72 (± 0.45)	83.25 (± 0.33)
	Partial	BIOES	86.18 (± 0.56)	**83.97** (± 0.38)
	Collobert's method		85.01 (± 0.31)	83.24 (± 0.26)
Both	Best combination		**87.77** (± 0.29)	83.82 (± 0.19)

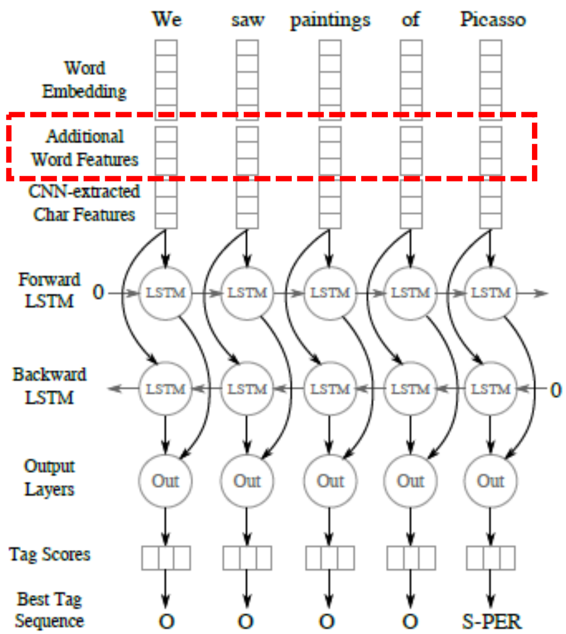
Table 9: Comparison of lexicon and matching/encoding methods over the BLSTM-CNN model employing random embeddings and no other features. When using both lexicons, the best combination of matching and encoding is Exact-BIOES for SENNA and Partial-BIOES for DBpedia. Note that the SENNA lexicon already contains "partial entries" so exact matching in that case is really just a more primitive form of partial matching.

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons

2. Data와 Lexicon feature 관점



Lexicon features

SENNA -> CoNLL에서 좋은 성능

Clean / Tailored to NW

DBpedia -> OntoNotes에서 좋은 성능

Noisier / Broader coverage

왜 그런가?

Lexicon	Matching	Encoding	CoNLL-2003	OntoNotes
No lexicon	-	-	83.38 (± 0.20)	82.53 (± 0.40)
SENNA	Exact	YN	86.21 (± 0.39)	83.24 (± 0.33)
	Exact	BIOES	86.14 (± 0.48)	83.01 (± 0.52)
DBpedia	Exact	YN	84.93 (± 0.30)	83.15 (± 0.26)
	Exact	BIOES	85.02 (± 0.23)	83.39 (± 0.39)
	Partial	YN	85.72 (± 0.45)	83.25 (± 0.33)
	Partial	BIOES	86.18 (± 0.56)	**83.97** (± 0.38)
	Collobert's method		85.01 (± 0.31)	83.24 (± 0.26)
Both	Best combination		**87.77** (± 0.29)	83.82 (± 0.19)

Table 9: Comparison of lexicon and matching/encoding methods over the BLSTM-CNN model employing random embeddings and no other features. When using both lexicons, the best combination of matching and encoding is Exact-BIOES for SENNA and Partial-BIOES for DBpedia. Note that the SENNA lexicon already contains "partial entries" so exact matching in that case is really just a more primitive form of partial matching.

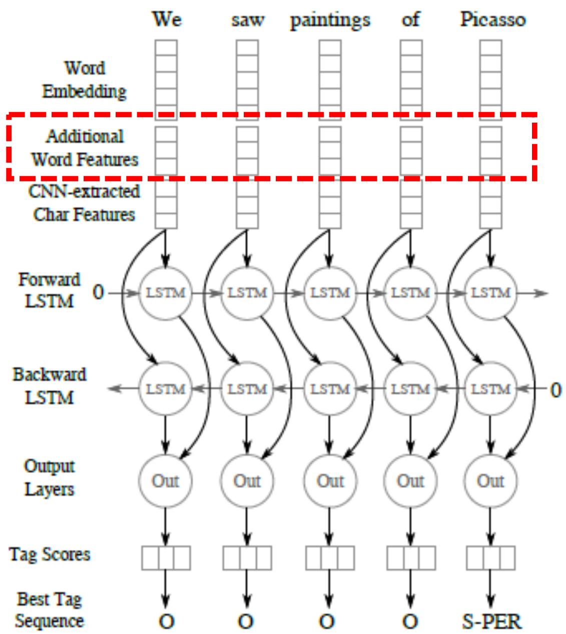
Contents

5.2 Core feature

5.2.2 Additional Word-level Features

Lexicons

2. Data와 Lexicon feature 관점



SENNA와 DBpedia로 만든 Lexicon과 Match
(흰색일 수록 Match가 잘 되는 것)
CoNLL은 OntoNotes보다 Lexicon매칭이 잘 됨
(CoNLL > OntoNotes)
Ex) CoNLL의 LOC Tag와 Lexicon의 LOC Tag

매칭이 잘 될 수록 성능에 도움됨 - Table 6 참조
Lexicon의 특성을 살리는 matching이 필요

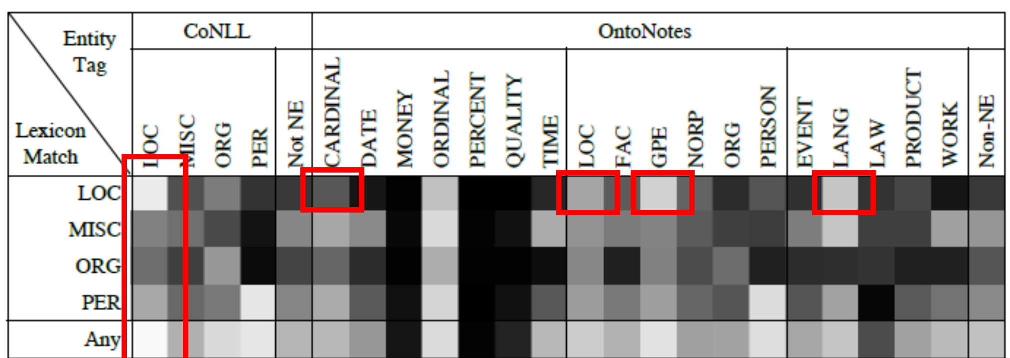


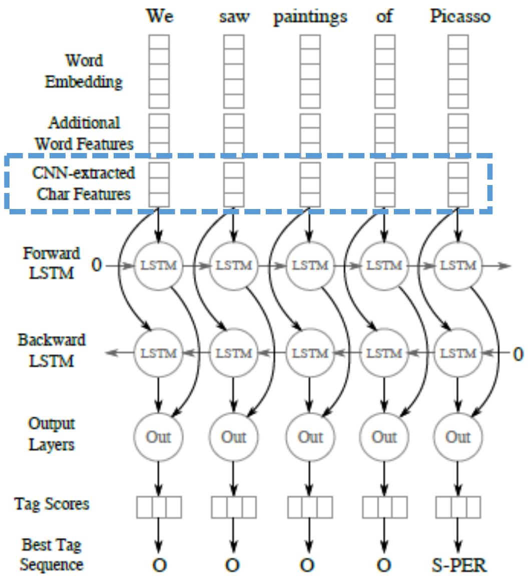
Figure 5: Fraction of named entities of each tag category matched completely by entries in each lexicon category of the SENNA/DBpedia combined lexicon. White = higher fraction.

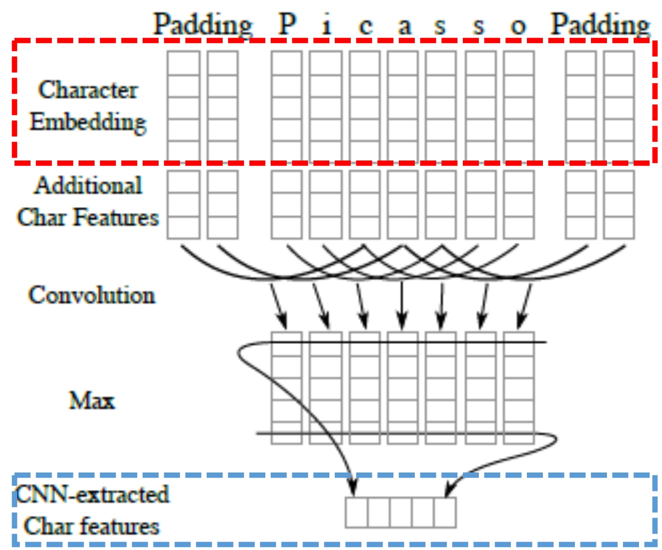
Contents

ACL 2016

5.2 Core feature

5.2.3 Character Embeddings





Lookup tables을 랜덤 값으로 초기화

균일분포의 [-0.5, 0.5] 사이의 값을 뽑아서 25차원의 character embedding 생성

CoNLL-2003에 포함된 character만 취급

PADDING과 UNKNOWN 토큰 추가

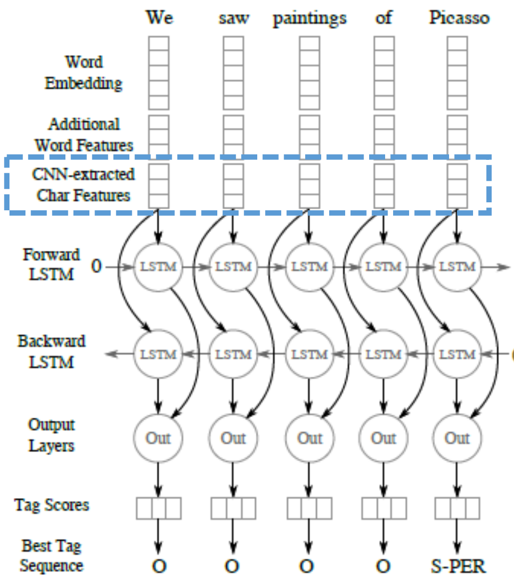
PADDING : CNN에 사용하는 padding

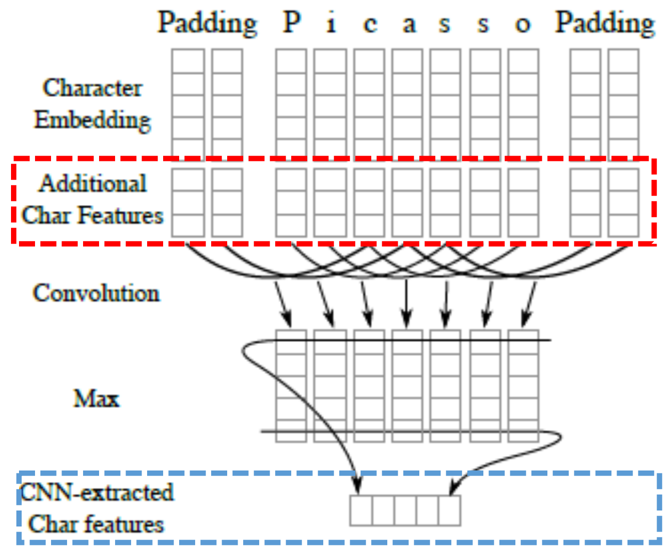
UNKNOWN : CoNLL-2003에는 없으나, OntoNotes에는 포함된 문자

Contents

5.2 Core feature

5.2.4 Additional Character-level Features





Lookup table을 사용하여, 문자의 종류를 나타내는 4 dimensional vector를 표현

- Upper case
- Lower case
- Punctuation
- Other

추가 feature들은 character-level CNN의 입력 값으로 사용될 뿐만 아니라, capitalization features와 비교할 때에도 사용

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$$logP([y]_1^T | [x]_1^T, \theta')$$

Implementation

Torch7 library

LSTM state의 초기값은 모두 0으로 설정
임베딩 제외한 Lookup table은 랜덤설정

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$$logP([y]_1^T|[x]_1^T,\theta')$$

Implementation

Torch7 library

Sentence-level log-likelihood가 maximize가 되도록 학습

Objective function

$$logP([y]_1^T|[x]_1^T,\theta') = S([x]_1^T|[y]_1^T,\theta') - log\sum_{\forall[j]_1^T} e^{s([x]_1^T|[j]_1^T,\theta')}$$

Contents

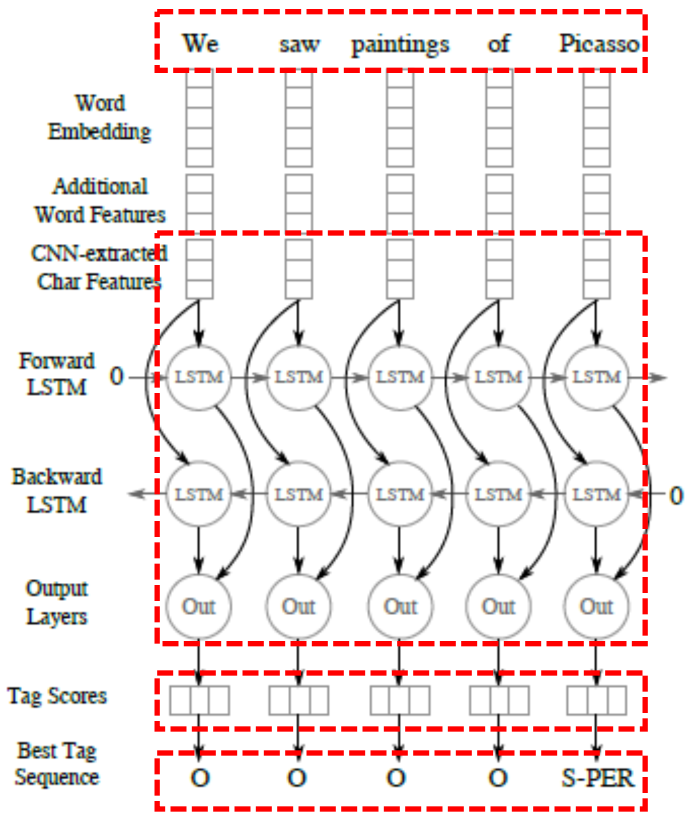
6.1 Score of a sequence of tag

Objective function (Softmax form)

$$\log P([y]_1^T \mid [x]_1^T, \theta') = S([x]_1^T \mid [y]_1^T, \theta') - \log \sum_{\forall [j]_1^T} e^{\boxed{s([x]_1^T \mid [j]_1^T, \theta')}}$$

Score of a sequence of tag $[i]_1^T$

$$S([x]_1^T, [i]_1^T, \theta') = \sum_{t=1}^{T} (A_{[i]_{t-1},[i]_t} + [f_\theta]_{[i]_t,t})$$



$[x]_1^T$

$$\theta' = \theta \cup \{A_{i,j}\ \forall i, j\}$$

$[f_\theta]_{i,t}$

$[i]_1^T$

$S([x]_1^T, [i]_1^T, \theta')$

6.1 Score of a sequence of tag

Score of a sequence of tag $[i]_1^T$

$$: S([x]_1^T, [i]_1^T, \theta') = \sum_{t=1}^{T} (A_{[i]_{t-1},[i]_t} + [f_\theta]_{[i]_t,t})$$

Length (number of word) of a sentence

All parameter to be trained

Score calculate from **neural-network**

Sequence of tag

Score calculate **from tag-transition matrix**

Given sentence

Contents

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$logP([y]_1^T | [x]_1^T, \theta')$

Implementation

Torch7 library

여러가지 Learning algorithm으로 실험

- Mini-batch SGD (ascent)
- Momentum
- AdaDelta
- RMSProp

Mini-batch SGD (ascent)의 성능이 좋아서 채택

고정된 학습률을 사용

Mini-batch 학습

각각의 mini-batch는 같은 길이(토큰 개수)를 가진 여러 문장들로 이루어짐

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

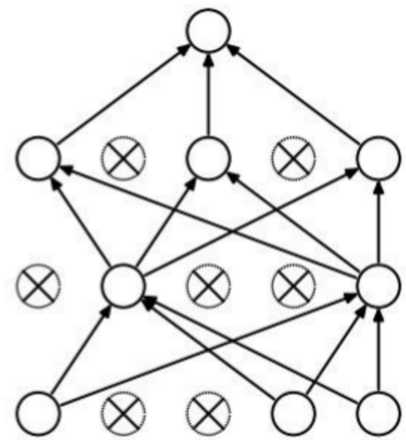
Objective function

$$logP([y]_1^T \,|\, [x]_1^T, \theta')$$

Implementation

Torch7 library

각각의 LSTM의 Output node에 적용
Overfitting을 줄이는데 효과적



Contents

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$$logP([y]_1^T | [x]_1^T, \theta')$$

Implementation

Torch7 library

각각의 LSTM의 Output node에 적용
Overfitting을 줄이는데 효과적

Dropout	CoNLL-2003		OntoNotes 5.0	
	Dev	Test	Dev	Test
-	93.72 (± 0.10)	90.76 (± 0.22)	82.02 (± 0.49)	84.06 (± 0.50)
0.10	93.85 (± 0.18)	90.87 (± 0.31)	83.01 (± 0.39)	84.94 (± 0.25)
0.30	94.08 (± 0.17)	91.09 (± 0.18)	83.61 (± 0.32)	85.44 (± 0.33)
0.50	94.19 (± 0.18)	91.14 (± 0.35)	84.35 (± 0.23)	**86.36** (± 0.28)
0.63	-	-	84.47 (± 0.23)	86.29 (± 0.25)
0.68	**94.31** (± 0.15)	**91.23** (± 0.16)	-	-
0.70	**94.31** (± 0.24)	91.17 (± 0.37)	**84.56** (± 0.40)	86.17 (± 0.25)
0.90	94.17 (± 0.17)	90.67 (± 0.17)	81.38 (± 0.19)	82.16 (± 0.18)

Table 8. F1 score results with dropout

Training data set 만 사용했을 때 실험 결과

나머지 Parameter는 Table 5와 같음

Hyper-parameter 튜닝전에는

0.63~0.68 값을 성능 향상을 위해 채택함

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$logP([y]_1^T | [x]_1^T, \theta')$

Implementation

Torch7 library

Hyper-parameter Optimization을 위해 2가지 방법 시도

각각 500가지 셋팅에 대해 실험함

Round	CoNLL-2003	OntoNotes 5.0
1	93.82 (± 0.15)	**84.57** (± 0.27)
2	**94.03** (± 0.23)	84.47 (± 0.29)

Table 4: Development set F1 score performance of the best hyper-parameter settings in each optimization round.

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$$logP([y]_1^T | [x]_1^T, \theta')$$

Implementation

Torch7 library

Round 1) Random search

각각 500가지 셋팅에 대해 실험함

Round	CoNLL-2003	OntoNotes 5.0
1	93.82 (± 0.15)	**84.57** (± 0.27)
2	**94.03** (± 0.23)	84.47 (± 0.29)

Table 4: Development set F1 score performance of the best hyper-parameter settings in each optimization round.

1) CoNLL-2003의 Development set에서 Random search 수행
2) OntoNotes 5.0 Development set에서 learning rate & epoch 만 튜닝

➡ OntoNotes 5.0에 더 적합

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$logP([y]_1^T | [x]_1^T, \theta')$

Implementation

Torch7 library

Round 2) Particle swarm

각각 500가지 셋팅에 대해 실험함

Round	CoNLL-2003	OntoNotes 5.0
1	93.82 (± 0.15)	**84.57** (± 0.27)
2	**94.03** (± 0.23)	84.47 (± 0.29)

Table 4: Development set F1 score performance of the best hyper-parameter settings in each optimization round.

1) Random search 보다 효율적임 (Clerc and Kennedy 2003)
2) Training이 가끔씩 실패함

➡ CoNLL-2003 에 더 적합

Contents

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Learning algorithm

Mini-batch SGD
Dropout

Objective function

$$logP([y]_1^T \mid [x]_1^T, \theta')$$

Implementation

Torch7 library

최종 Hyper-parameter setting

Hyper-parameter	CoNLL-2003 (Round 2)		OntoNotes 5.0 (Round 1)	
	Final	Range	Final	Range
Convolution width	**3**	[3, 7]	**3**	[3, 9]
CNN output size	**53**	[15, 84]	**20**	[15, 100]
LSTM state size	**275**	[100, 500]	**200**	[100, 400][10]
LSTM layers	**1**	[1, 4]	**2**	[2, 4]
Learning rate	**0.0105**	$[10^{-3}, 10^{-1.8}]$	**0.008**	$[10^{-3.5}, 10^{-1.5}]$
Epochs[11]	**80**	-	**18**	-
Dropout[12]	**0.68**	[0.25, 0.75]	**0.63**	[0, 1]
Mini-batch size	**9**	-[13]	**9**	[5, 14]

Table 3: Hyper-parameter search space and final values used for all experiments

Epoch 수는 Overfitting을 고려하여 결정

6. Training

Hyper-parameter Optimization

Random search
Particle swarm

Training 후 Inference

Viterbi algorithm

$[i]_1^T$

Learning algorithm

Mini-batch SGD
Dropout

Mini-batch SGD, fixed learning rate 사용
Dropout, LSTM의 Output node에 적용

Objective function

$$logP([y]_1^T | [x]_1^T, \theta')$$

Maximize Sentence level log-likelihood

Implementation

Torch7 library

LSTM state의 초기값은 모두 0으로 설정
임베딩 제외한 Lookup table은 랜덤설정

6. Training

6.2 Excluding Failed Trials

Failed Trials

BLSTM: 문제없이 학습 성공
BLSTM-CNN: 실패 발생
CoNLL 2003: 5~10%
Ontonotes: 1.5%

Successful 기준

CoNLL-2003
- Dev set에서 F1 < 95 제외

OntoNotes 5.0
- Training set에서 F1 < 80 제외

10 successful ones 채택

극복 방법들

1) Using a lower learning rate
2) Clipping gradients
3) AdaDelta(크게 이득없음)

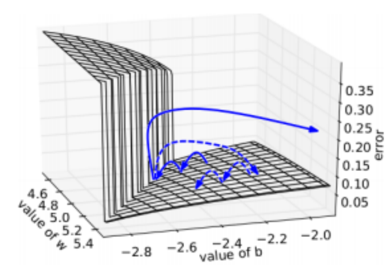


Figure 5: Gradient explosion clipping visualization

Contents

7. Result

Baseline

Emb 효과

Model	CoNLL-2003			OntoNotes 5.0		
	Prec.	Recall	F1	Prec.	Recall	F1
FFNN + emb + caps + lex	89.54	89.80	89.67 (± 0.24)	74.28	73.61	73.94 (± 0.43)
BLSTM	80.14	72.81	76.29 (± 0.29)	79.68	75.97	77.77 (± 0.37)
BLSTM-CNN	83.48	83.28	83.38 (± 0.20)	82.58	82.49	82.53 (± 0.40)
BLSTM-CNN + emb	90.75	91.08	90.91 (± 0.20)	85.99	86.36	86.17 (± 0.22)
BLSTM-CNN + emb + lex	91.39	**91.85**	**91.62** (± 0.33)	**86.04**	**86.53**	**86.28** (± 0.26)
Collobert et al. (2011b)	-	-	88.67	-	-	-
Collobert et al. (2011b) + lexicon	-	-	89.59	-	-	-
Huang et al. (2015)	-	-	90.10	-	-	-
Ratinov and Roth (2009)[18]	91.20	90.50	90.80	82.00	84.95	83.45
Lin and Wu (2009)	-	-	90.90	-	-	-
Finkel and Manning (2009)[19]	-	-	-	84.04	80.86	82.42
Suzuki et al. (2011)	-	-	91.02	-	-	-
Passos et al. (2014)[20]	-	-	90.90	-	-	82.24
Durrett and Klein (2014)	-	-	-	85.22	82.89	84.04
Luo et al. (2015)[21]	**91.50**	91.40	91.20	-	-	-

Table 5: Results of our models, with various feature sets, compared to other published results. The three sections are, in order, our models, published neural network models, and published non-neural network models. For the features, emb = Collobert word embeddings, caps = capitalization feature, lex = lexicon features from both SENNA and DBpedia lexicons. For F1 scores, standard deviations are in parentheses.

Baseline인 FFNN과 기존 모델들을 능가하는 성능을 보임

BLSTM-CNN + emb + lex Model의 성능이 가장 뛰어남

제안된 모델은 자동으로 NER에 맞는 feature를 학습함

(feature engineering 대체가능)

Contents

7. Result

Features	BLSTM		BLSTM-CNN		BLSTM-CNN + lex	
	CoNLL	OntoNotes	CoNLL	OntoNotes	CoNLL	OntoNotes
none	76.29 (± 0.29)	77.77 (± 0.37)	83.38 (± 0.20)	82.53 (± 0.40)	87.77 (± 0.29)	83.82 (± 0.19)
emb	88.23 (± 0.23)	82.72 (± 0.23)	90.91 (± 0.20)	86.17 (± 0.22)	**91.62** (± 0.33)	86.28 (± 0.26)
emb + caps	90.67 (± 0.16)	86.19 (± 0.25)	90.98 (± 0.18)	86.35 (± 0.28)	91.55 (± 0.19)*	86.28 (± 0.32)*
emb + caps + lex	**91.43** (± 0.17)	**86.21** (± 0.16)	**91.55** (± 0.19)*	86.28 (± 0.32)*	91.55 (± 0.19)*	86.28 (± 0.32)*
emb + char	-	-	90.88 (± 0.48)	86.08 (± 0.40)	91.44 (± 0.23)	**86.34** (± 0.18)
emb + char + caps	-	-	90.88 (± 0.31)	**86.41** (± 0.22)	91.48 (± 0.23)	86.33 (± 0.26)

Table 6: F1 score results of BLSTM and BLSTM-CNN models with various additional features; emb = Collobert word embeddings, char = character type feature, caps = capitalization feature, lex = lexicon features. Note that starred results are repeated for ease of comparison.

CNN은 hand-crafted
character feature **대체가능성** 보임

F1 score

BLSTM < BLSTM-CNN, + lex

OntoNotes에서는 성능에 도움됨
그러나 CoNLL에서는 성능 저하시킴

Char + Caps feature

CoNLL < OntoNotes

Lexicon feature는 성능에 도움이 되나
CoNLL처럼 잘 Match 될때만 해당됨

Lexicon feature

OntoNotes < CoNLL

Contents

7. Result

7.2 Analysis of OntoNotes Performance

Clean text

Broadcast news (BN)
Newswire (NW)

Noisy text

Telephone conversation (TC)
Web text (WB)

Clean text 에서 Best Performance
Noisy text 에서 Worst Performance

Model	BC	BN	MZ	NW	TC	WB
Test set size (# tokens)	32,576	23,557	18,260	51,667	11,015	19,348
Test set size (# entities)	1,697	2,184	1,163	4,696	380	1,137
Finkel and Manning (2009)	78.66	87.29	82.45	85.50	67.27	72.56
Durrett and Klein (2014)[38]	78.88	87.39	82.46	87.60	**72.68**	76.17
BLSTM-CNN	81.26	86.87	79.94	85.27	67.82	72.11
BLSTM-CNN + emb	85.05	**89.93**	84.31	88.35	72.44	77.90
BLSTM-CNN + emb + lex	**85.23**	**89.93**	**84.45**	**88.39**	72.39	**78.38**

Table 10: Per genre F1 scores on OntoNotes. BC = broadcast conversation, BN = broadcast news, MZ = magazine, NW = newswire, TC = telephone conversation, WB = blogs and newsgroups

Contents

8. Conclusion

Conclusion & Contribution

1) NER분야에서 BLSMT-CNN 모델의 우수성을 보임
   Feature Engineering 최소화
2) Partial matching lexicon algorithm으로
   NER 성능 향상 가능을 보임
3) Word embedding의 Training data의 Domain이 중요

Future work

Extended tagset NER, Entity linking