End to end Sequence Labeling via Bi directional LSTM CNNs CRF

1. Introduction

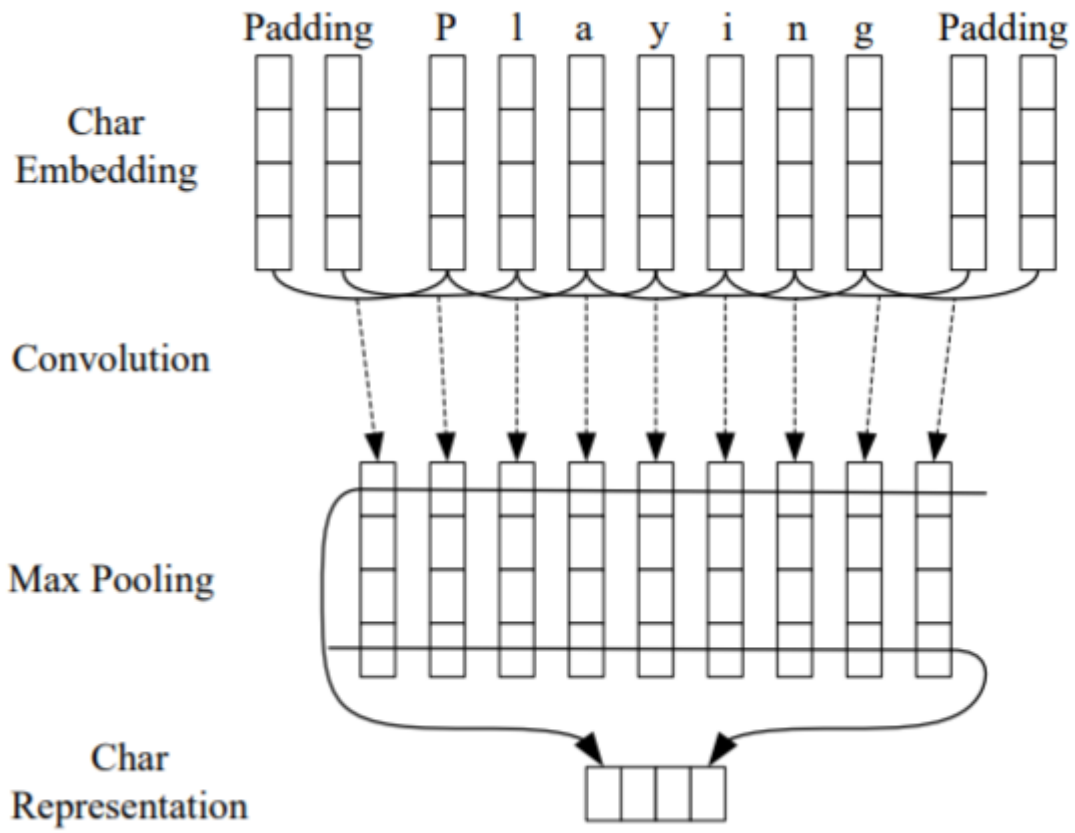
- 완전히 end-to-end model이다.
- 그래서 feature engineering도 필요없고 data 전처리도 필요없다!
- 우선 단어의 character-level의 정보를 character-level의 representation으로 인코딩하기 위해 CNN을 사용했다.
- 그리고 character-level과 word-level의 representation을 섞고 bi-directional LSTM에 feed했다. 이렇게 하면 각각의 단어의 맥락 정보를 모델링할 수 있다고 한다.
- BLSTM의 제일 위에 전체 문장의 label들을 디코드하기 위해 sequential CRF를 사용했다.
- POS tagging과 named entity recognition(NER)두 개의 labeling task로 평가를 했다.

2. Neural Network Architecture

- bottom부터 top까지 차례차례 소개하겠다.

2.1. CNN for Character-level Representation

- CNN은 형태론적인 정보를 추출하는데 효과적이다.(어간, 어미...)



-dashed arrows: dropout layer

- character-level representation을 뽑아냈다.
- character embedding만 CNN의 input으로 사용했다.
- character embedding이 CNN의 input이 되기 전에 dropout layer를 사용했다.

2.2. Bi-directional LSTM

- 많은 sequence labeling task들에서는 이전 context와 이후 context를 모두 이용하는 게 좋다.
- 하지만 LSTM의 hidden state h_t는 이전 context에서만 정보를 가져온다.
- 이러한 단점을 보완한게 바로 Bi-directional LSTM이다.
- 기본적인 아이디어는 각각의 forward hidden state와 backward hidden state를 이용해 이전과 이후 정보를 이용하자는 것이다.
  두 개의 hidden state는 final output을 위해 concatenate 된다.

2.3. CRF

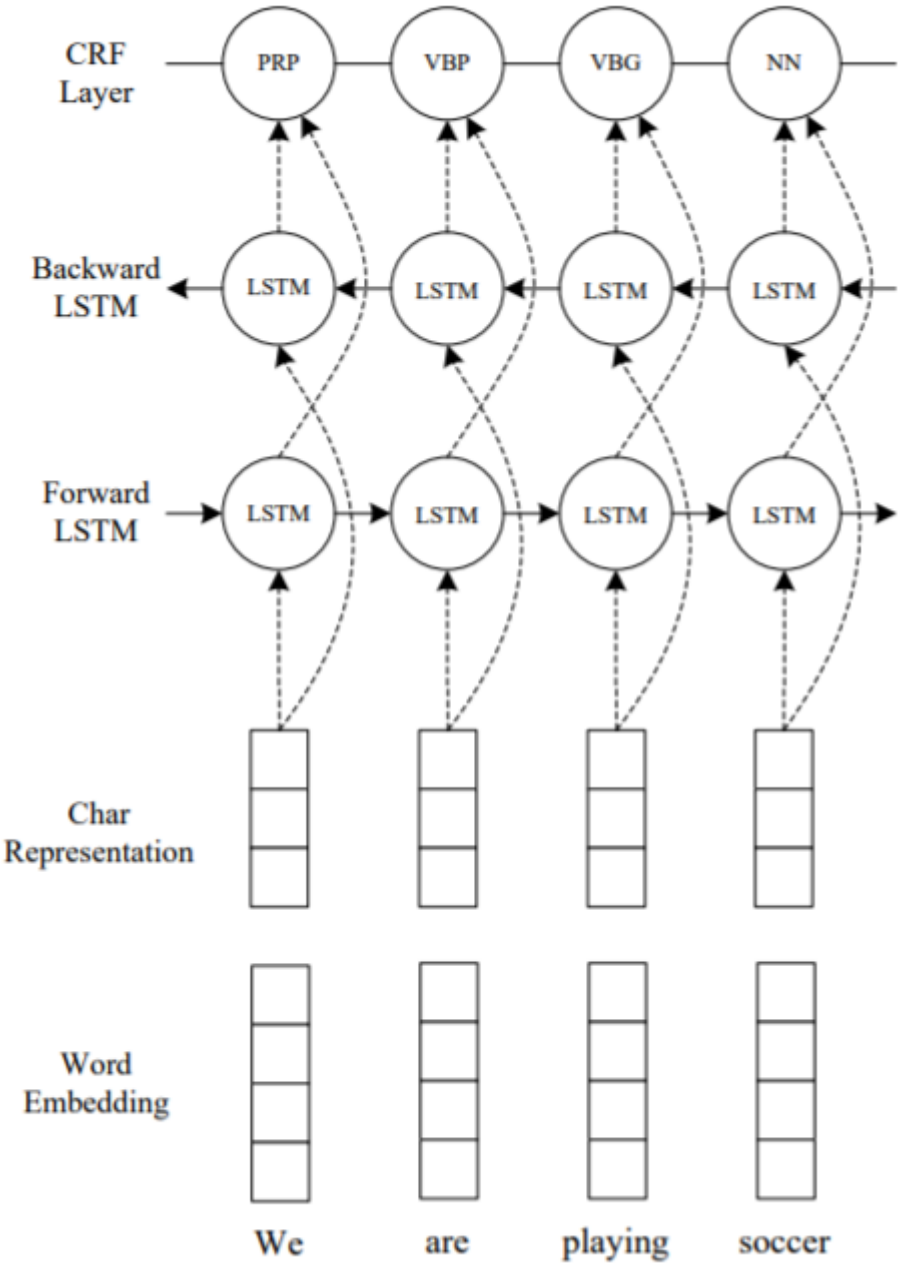
- conditional random field의 약자
- 각각의 label을 독립적으로 디코드 하는게 아니라, 주변을 보고 labeling을 한다고 보면 될 것 같다.
- 주어진 input sequence에 대해 주변부를 보고 제일 좋은 label chain을 찾는 것이다.
- 예를 들어, POS tagging에서 명사 다음에 동사보다는 형용사가 올 확률이 높다. 그렇다면 label chain은 명사-동사가 아닌 명사-형용사가 더 좋을 것이다.
- z는 일반적인(generic) input sequence를 나타내고, z = {z_1, ..., z_n} 이다. z_i는 i번째 단어에 대한 input vector 이다.
- y는 z에 대한 label의 일반적인 sequence이다. y = {y_1, ..., y_n} (위와 같이 input이 주어졌을 때, 보통의 label)
- y(z)는 z에 대한 가능한 label sequence의 집합을 나타낸다.
- z가 주어졌을 때, 모든 가능한 label sequence y에 대한 conditional probability는 다음과 같이 정의된다.

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

- $\psi_i(y', y, z) = exp(W^T_{y',y}z_i + b_{y',y})$
- $W^T_{y',y}$와 $b_{y',y}$는 label pair (y',y)에 대한 weight vector와 bias
- CRF 학습을 위해 maximum conditional likelihood estimation을 쓴다.
- training set (z_i, y_i)에 대한 log-likelihood는 다음과 같다.
  L(W,b) = $\sum_i logp(y|z;W,b)$
- maximum likelihood training은 L(W,b)가 최대가 되는 parameter들을 선택한다.
- decoding은 가장 높은 conditional probability로 label sequence y*을 찾는 것이다.
  y* = $argmax_{y\in y(z)}p(y|z;W,b)$
- CRF model은 두 개의 연속적인 label들 간의 interaction만 고려한다.
- Viterbi 알고리즘을 사용해서 효율적으로 decoding한다. Viterbi 알고리즘

2.4. BLSTM-CNNs-CRF

- BLSTM의 output vector들을 CRF layer에 feed해서 model을 만들었다.



- 각각의 단어에 대해, character-level representation은 CNN에 char embeddings을 input으로 넣어서 계산됐다.(2.1. 참고)
- character-level representation vector는 BLSTM에 feed되기 위해 word embedding vector와 concatenate된다. (?)
- 가장 좋은 label sequence를 디코드하기 위해 BLSTM의 output vector들은 CRF layer로 feed된다.
- 위의 그림에서도 보여지듯이, dropout layer는 BLSTM의 input과 output vector 모두에 적용된다.
- 실험에서 dropout을 사용하는 것이 상당히 model의 성능을 향상했다.

3. Network Training

- neural network를 학습시키는데의 detail을 제공하겠다.

3.1. Parameter Initialization

Word Embeddings

- 각각 다른 set으로 GloVe 100-dim embeddings, Senna 50-dim embeddings, Word2Vec 300-dim을 사용했다. (pre-trained embedding을 사용했다는 뜻인 듯)
- pretrained word embeddings의 효과를 검증하기 위해 random하게 초기화된 100-dim embeddings도 사용했다. (uniformly sampled from range $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$)

Character Embeddings

- uniformly sampled from range $[-\sqrt{\frac{3}{dim}}, +\sqrt{\frac{3}{dim}}]$
- dim은 30으로 설정했다.

Weight Matrices and Bias Vectors

- matrix parameter들은 $[-\sqrt{\frac{6}{r+c}}, +\sqrt{\frac{6}{r+c}}]$에서 uniform sample로 random하게 초기화됐다.(r과 c는 행과 열의 개수)
- LSTM의 forget gate의 bias b_f만 1.0으로 초기화 하고, 나머지 bias vector들은 0으로 초기화했다.

3.2. Optimization Algorithm

- batch size 10에 momentum 0.9로 mini batch SGD를 사용했다.
- 초기 학습율은 POS tagging은 0.01, NER은 0.015로 했다.
- training의 각 epoch당 lr = lr/(1+p_t)로 학습율을 업데이트 했다. (decay rate p=0.05, t는 epoch 수)
- gradient exploding 효과를 줄이기 위해 gradient clipping을 사용했다.
- 다른 최적화 알고리즘도 써봤는데 SGD with momentum과 gradient clipping보다 좋은 게 없었다.

Early Stopping

- 최적의 parameter들이 나타났을 때 멈췄다.

Fine Tuning

- 각각의 embeddings에 대해 gradient updates동안 변경하면서 fine-tune했다.

Dropout Training

- overfitting을 완화하기 위해 사용했다.
- CNN에 input으로 넣기 전 character embeddings와 BLSTM의 input과 output vector 모두에 적용했다.
- dropout rate를 0.5로 고정시켰다.
- dropout을 써서 model에 상당한 향상이 있었다.

4. Experiments

4.1. Data Sets

POS Tagging

- Wall Street Journal (WSJ) portion of Penn Treebank (PTB): 45개의 다른 POS tag들을 포함하고 있다. (English)

  NER
- CoNLL 2003 shared task: PERSON, LOCATION, ORGANIZATION, MISC의 4개의 다른 타입의 개체명을 포함하고 있다.(English)

- standard BIO2 대신 BIOES tagging scheme을 사용했다.
  data set에 전혀 전처리를 하지 않았다.

4.2. Main Results

	POS		NER					
	Dev	Test	Dev			Test		
Model	Acc.	Acc.	Prec.	Recall	F1	Prec.	Recall	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

- 모두 GloVe 100-dim word embeddings를 사용했다.
- BLSTM이 BRNN보다 두 가지 task모두에서 나았다.
- BLSTM-CNN model은 상당히 BLSTM model을 뛰어넘었다. (언어적 sequence labeling task에서는 char-level representation이 중요하다는 것을 보여준다.)
- 마지막으로 CRF layer를 더하니 성능이 BLSTM-CNN보다 상당히 향상됐다.

4.4 Word Embeddings

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**

- random embeddings를 쓰는 것보다 pretrained word embeddings를 쓰는 게 상당한 향상을 이뤄냈다.
- NER이 훨씬 더 pretrained embeddings에 영향을 많이 받았다.

5. Conclusion

- sequence labeling을 하는 완전한 end-to-end model이다.
- 우리의 model은 multi-task learning 접근법을 통해 더 발전될 수 있을 것이다.