Semi-supervised sequence tagging with bidirectional language models

WeiYang

weiyang@godweiyang.com

www.godweiyang.com

East China Normal University
Department of Computer Science and Technology

2017.11.03

Outline

Backgrounds

- ubiquitous pre-trained word embedding
- context are also essential
  "A Central Bank spokesman" vs "The Central African Republic"
- previous work learned the bi-RNN from supplemental labeled data
  - transfer learning
- alternative
  - neural language model pre-trained on unlabeled corpus
  - supervised sequence tagging model

Contributions

- the context sensitive representation captured in the LM embeddings is useful in the supervised sequence tagging setting
- using both forward and backward LM embeddings boosts performance over a forward only LM
- the LM needn't to transfer across domains

Overview

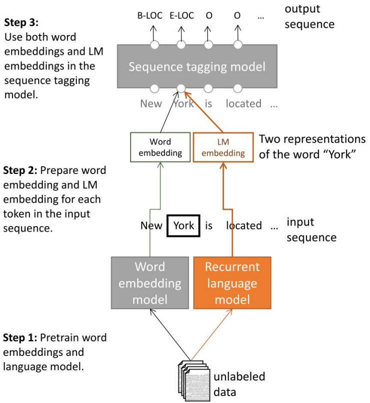

Figure: The main components in TagLM

Overview

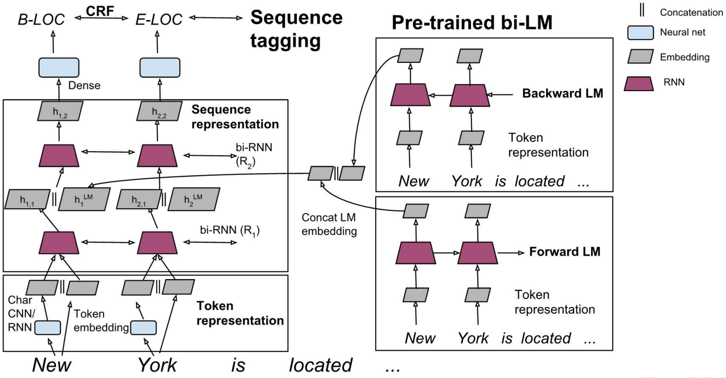

Figure: Overview of TagLM

Baseline sequence tagging model

- sentence

$$(t_1, t_2, \ldots, t_N) \tag{1}$$

- token representation: x_k , character based representation: c_k , token embedding: w_k where

$$\begin{aligned} c_k &= C(t_k; \theta_c) \\ w_k &= E(t_k; \theta_w) \\ x_k &= [c_k; w_k] \end{aligned} \tag{2}$$

Baseline sequence tagging model

- For each token position, k, the hidden state $h_{k,i}$ of RNN layer i is formed by concatenating the hidden states from the forward($\vec{h}_{k,i}$) and backward($\overleftarrow{h}_{k,i}$) RNNs
- $$\begin{aligned} \vec{h}_{k,1} &= \vec{R}_1(x_k, \vec{h}_{k-1,1}; \theta_{\vec{R}_1}) \\ \overleftarrow{h}_{k,1} &= \overleftarrow{R}_1(x_k, \overleftarrow{h}_{k+1,1}; \theta_{\overleftarrow{R}_1}) \\ h_{k,1} &= [\vec{h}_{k,1}; \overleftarrow{h}_{k,1}] \end{aligned} \tag{3}$$
- usually $L = 2$

Baseline sequence tagging model

- predict a score for each possible tag using a single dense layer with $h_{k,L}$
- model and decode each sentence jointly instead of independently predicting the label for each token
    - CRF and Viterbi

Bidirectional LM

- a language model computes the probability of a token sequence $(t_1, t_2, \ldots, t_N)$

$$p(t_1, t_2, \ldots, t_N) = \prod_{k-1}^{N} p(t_k | t_1, t_2, \ldots, t_{k-1}) \quad (4)$$

- previous work only have forward LM embedding $\vec{h}_k^{LM}$
- add a backward LM in additional to the traditional forward LM

$$p(t_1, t_2, \ldots, t_N) = \prod_{k-1}^{N} p(t_k | t_{k+1}, t_{k+2}, \ldots, t_N) \quad (5)$$

Bidirectional LM

- use two LSTMs to pre-training the forward and backward LMs separately
- the forward and backward LMs are independent, without any shared parameters
- final LM embeddings

$$h_k{}^{LM} = [\vec{h}_k^{LM}; \overleftarrow{h}_k{}^{LM}] \tag{6}$$

Combining LM with sequence model

- introducing the LM embeddings at the output of the first layer, replace (2) with

$$h_{k,1} = [\overrightarrow{h}_{k,1}; \overleftarrow{h}_{k,1}; {h_k}^{LM}] \tag{7}$$

- many possibilities for adding the LM embeddings to the sequence model
  - non-linear mapping, replace (6) with

$$h_{k,1} = f([\overrightarrow{h}_{k,1}; \overleftarrow{h}_{k,1}; {h_k}^{LM}]) \tag{8}$$

  - attention-like mechanism

Overview

- two sequence tagging tasks
  - CoNLL 2003 NER
  - CoNLL 2000 chunking
- BIOES labeling scheme
- Senna word embeddings
- pre-processed the text
  - lowercase all tokens
  - replace all digits with 0

Pre-trained language models

- 1B Word Benchmark
- forward LSTM-2048-512 and backward LSTM-2048-512
- synchronous parameter updates across four GPUs

Training

1. first train with a constant learning rate $\alpha = 0.001$
2. then at the epoch with the highest development performance, decrease α an order of magnitude
3. train for five epochs
4. decrease α an order of magnitude
5. train for five epochs
6. stop

Overall system results

Model	$F_1 \pm$ std
Chiu and Nichols (2016)	90.91 ± 0.20
Lample et al. (2016)	90.94
Ma and Hovy (2016)	91.37
Our baseline without LM	90.87 ± 0.13
TagLM	$\mathbf{91.93 \pm 0.19}$

Table 1: Test set F_1 comparison on CoNLL 2003 NER task, using only CoNLL 2003 data and unlabeled text.

Overall system results

Model	$F_1 \pm$ std
Yang et al. (2017)	94.66
Hashimoto et al. (2016)	95.02
Søgaard and Goldberg (2016)	95.28
Our baseline without LM	95.00 ± 0.08
TagLM	$\mathbf{96.37 \pm 0.05}$

Table 2: Test set F_1 comparison on CoNLL 2000 Chunking task using only CoNLL 2000 data and unlabeled text.

Overall system results

Model	**External resources**	F_1 **Without**	F_1 **With**	Δ
Yang et al. (2017)	transfer from CoNLL 2000/PTB-POS	91.2	91.26	+0.06
Chiu and Nichols (2016)	with gazetteers	90.91	91.62	+0.71
Collobert et al. (2011)	with gazetteers	88.67	89.59	+0.92
Luo et al. (2015)	joint with entity linking	89.9	91.2	+**1.3**
Ours	no LM vs TagLM *unlabeled data only*	90.87	**91.93**	+1.06

Table 3: Improvements in test set F_1 in CoNLL 2003 NER when including additional labeled data or task specific gazetteers (except the case of TagLM where we do not use additional labeled resources).

Overall system results

Model	External resources	F_1 Without	F_1 With	Δ
Yang et al. (2017)	transfer from CoNLL 2003/PTB-POS	94.66	95.41	+0.75
Hashimoto et al. (2016)	jointly trained with PTB-POS	95.02	95.77	+0.75
Søgaard and Goldberg (2016)	jointly trained with PTB-POS	95.28	95.56	+0.28
Ours	no LM vs TagLM *unlabeled data only*	95.00	**96.37**	**+1.37**

Table 4: Improvements in test set F_1 in CoNLL 2000 Chunking when including additional labeled data (except the case of TagLM where we do not use additional labeled data).

How to use LM embeddings?

Use LM embeddings at	$F_1 \pm$ **std**
input to the first RNN layer	91.55 ± 0.21
output of the first RNN layer	$\mathbf{91.93 \pm 0.19}$
output of the second RNN layer	91.72 ± 0.13

Table 5: Comparison of CoNLL-2003 test set F_1 when the LM embeddings are included at different layers in the baseline tagger.

Does it matter which language model to use?

Forward language model	Backward language model	LM perplexity Fwd	LM perplexity Bwd	$F_1 \pm$ std
—	—	N/A	N/A	90.87 ± 0.13
LSTM-512-256*	LSTM-512-256*	106.9	104.2	90.79 ± 0.15
LSTM-2048-512	—	47.7	N/A	91.40 ± 0.18
LSTM-2048-512	LSTM-2048-512	47.7	47.3	91.62 ± 0.23
CNN-BIG-LSTM	—	30.0	N/A	91.66 ± 0.13
CNN-BIG-LSTM	LSTM-2048-512	30.0	47.3	$\mathbf{91.93 \pm 0.19}$

Table 6: Comparison of CoNLL-2003 test set F_1 for different language model combinations. All language models were trained and evaluated on the 1B Word Benchmark, except LSTM-512-256* which was trained and evaluated on the standard splits of the NER CoNLL 2003 dataset.

Other analysis

- Importance of task specific RNN
- Dataset size
- Number of parameters
- Does the LM transfer across domains?

Related work

- Unlabeled data
- Neural language models
- Interpreting RNN states
- Other sequence tagging models

Conclusion

- proposed a simple and general semi-supervised method using pre-trained neural language models to augment token representations in sequence tagging models
- add a backward LM in addition to traditional forward LMs consistently improves performance
- the LM needn't to transfer across domains
- even when the baseline model is trained on a large number of labeled examples, the performence can also get a large improvement