CkipTagger

2019/09/11

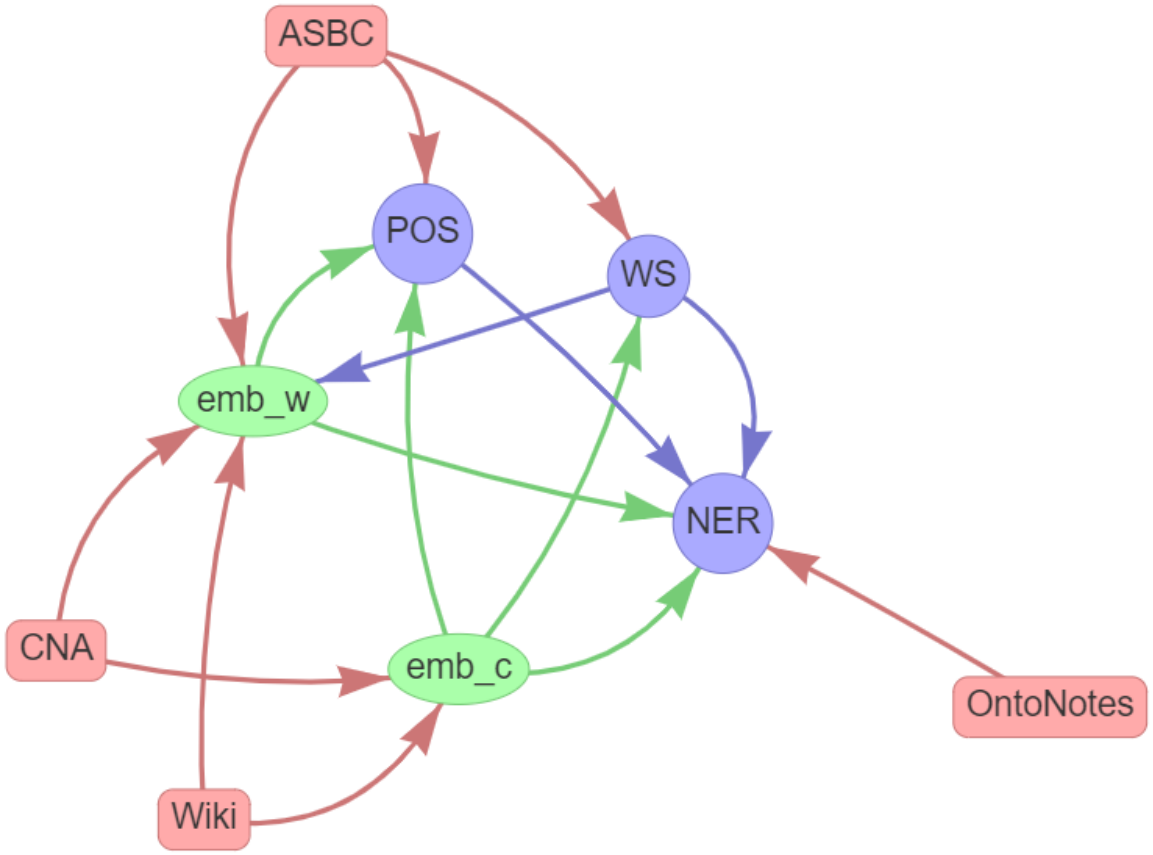
Peng-Hsuan Li

https://github.com/ckiplab/ckiptagger

Peng-Hsuan Li, Tsu-Jui Fu, WeiYun Ma. 2019. Remedying BiLSTM-CNN Deficiency in Modeling Cross-Context for NER. arXiv:1908.11046.

Roadmap

- Corpora
- Embedding
- Tools



Corpora

- Normalize
  - Unicode-normalization
  - Transform to ZhTW

- Datasets
  - CNA: Chinese Gigaword 5, CNA part
  - Wiki: Chinese wiki, 2019-05-20 pages-articles dump
  - ASBC: ASBC 4.0
  - OntoNotes: OntoNotes 5.0, Chinese part

	Sentences	Words	Characters	word/sent	char/sent	Sentence type
CNA	13,366,581	632,289,913	1,098,546,752	47.3	82.2	Paragraph
Wiki	5,557,141	247,714,633	461,862,002	44.6	83.1	Paragraph
ASBC	1,297,793	10,409,751	16,331,383	8.0	12.6	Clause
OntoNotes	46,905	958,345	1,515,151	20.4	32.3	Sentence

Embedding

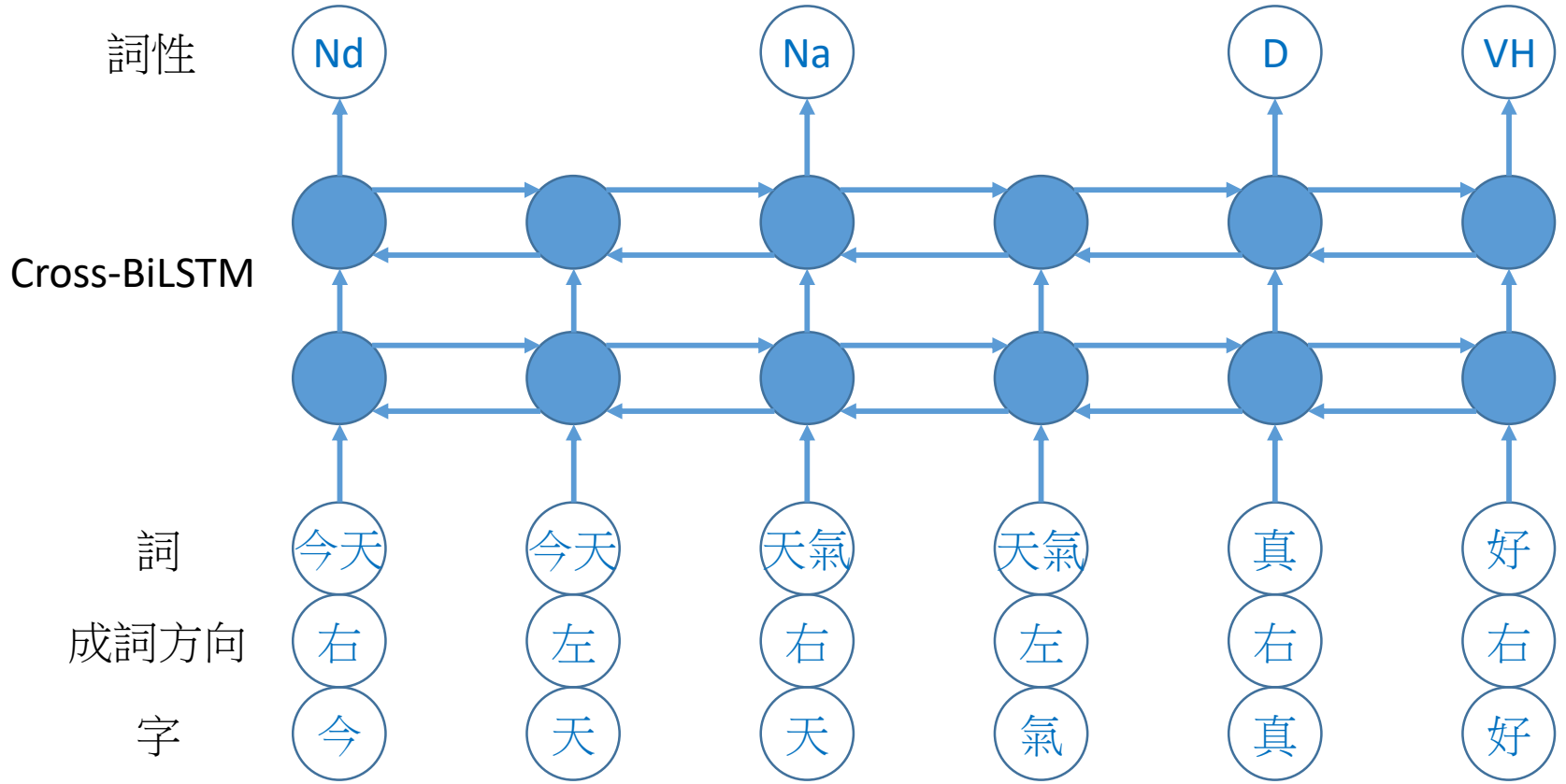
- Character
  - CNA + Wiki
  - 1,560,408,754 characters
  - [13136, 300]

- Word
  - CNA + Wiki + ASBC-train, segmented by WS
  - 890,414,297 words
  - [1355791, 300]
    - Excluding non-most frequent 30K and length>20

Tools

- WS
  - BiLSTM
  - Character-based
  - BI tagging
- POS
  - BiLSTM
  - Character-based with WS info
  - Only tag B characters
- NER
  - BiLSTM
  - Character-based with WS, POS info
  - BIOES tagging

POS Model



CkipTagger vs. CKIPWS

- Performance
  - WS: +1.4% absolute F1 on ASBC 4.0 test split
  - POS: +4.0% absolute accuracy on ASBC 4.0 test split

- Ease-of-use
  - Do not auto delete/change/add characters
    - E.g. Keep spaces as they are
    - E.g. Keep full/half-width characters as they are
    - E.g. Do not auto insert newlines
  - Support indefinitely long sentences

- Features
  - Do not rely on word list, word frequency statistics, POS frequency statistics
  - Support user-defined recommended-word list
  - Support user-defined must-word list

WS, POS Performance

- $POS\ acc = \frac{\#\ predicted\ words\ with\ correct\ boundary\ and\ POS\ tags}{\#\ gold\ standard\ words}$

	ASBC-test			
	WS			POS
	Prec.	Rec.	F1	Acc.
Gold WS + CkipTagger POS	--	--	--	97.20
CkipTagger WS+POS	97.49	97.17	97.33	94.59
CKIPWS	95.85	95.96	95.91	90.62
Jieba_zhTW	90.51	89.10	89.80	--

WS Performance with Dictionary

- Performance with target domain-specific dictionary
  - Preprocess: -0.2%
  - Hard post-process: -0.3%
  - Soft post-process: +0.2%

	ASBC-validate		
	Prec.	Rec.	F1
No dictionary	97.52	97.12	97.32
Preprocess	97.52	96.68	97.10
Hard post-process	97.17	96.90	97.03
soft post-process	97.80	97.21	97.50

- The final tool allows
  - A combination of hard + soft
  - Setting weights for each dictionary word

WS Speed

- Speed
  - (GPU) GTX 1080 Ti + (CPU) Xeon E5-2620 v4, using 3 cores

	Sentences	Words	Characters	word/sent	char/sent	sent/sec	word/sec	char/sec	Sentence Type
CNA	13,366,581	632,289,913	1,098,546,752	47.3	82.2	477	22,543	39,167	Paragraph
Wiki	5,557,141	247,714,633	461,862,002	44.6	83.1	410	18,271	34,066	Paragraph
ASBC-train	1,297,793	10,409,751	16,331,383	8.0	12.6	5,150	41,309	64,807	Clause

NER Performance

	Features	Embedding	OntoNotes Validate	OntoNotes Test
Old NER	CKIPWS	Morris	75.22	75.79
CkipTagger NER	CkipTagger WS+POS	CkipTagger	78.49	77.98