Sentence-State LSTM for Text Representation

Bidirectorial LSTMがもつ問題を、各wordに対してpararell stateを保持することにより解決。

https://github.com/leuchine/S-LSTM/blob/master/sequence_tagging/model/base_model.py

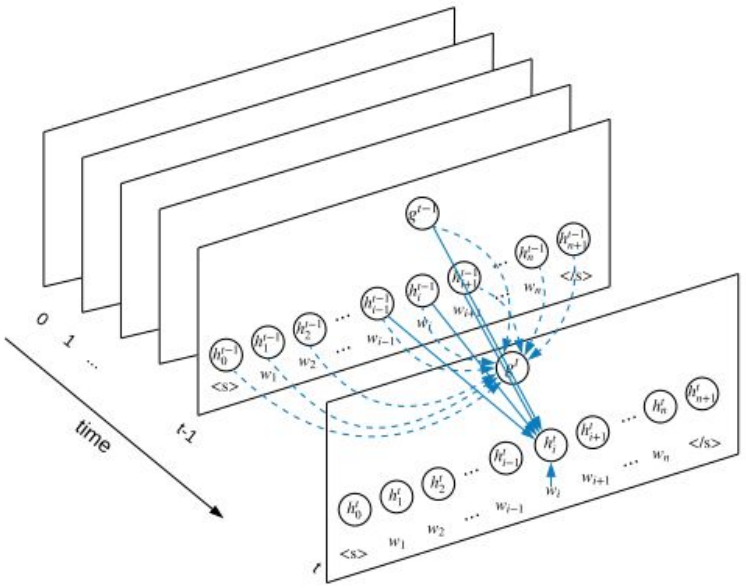


Figure 1: Sentence-State LSTM

shown in Figure 1, the main idea is to model the hidden states of all words simultaneously at each recurrent step, rather than one word at a time. In mation (Sabour et al., 2017). In contrast, S-LSTM uses a global sentence-level node to assemble and back-distribute local information in the recurrent state transition process, suffering less information loss compared to pooling.

Sentence-State LSTM for Text Representation

baselineとなるBi-LSTMのモデル

$\boldsymbol{s}: < s >, w_1, w_2, \ldots w_i, \ldots . w_n, </s >$

$\boldsymbol{h_i}$: hidden vector for each input word $\boldsymbol{w_i}$

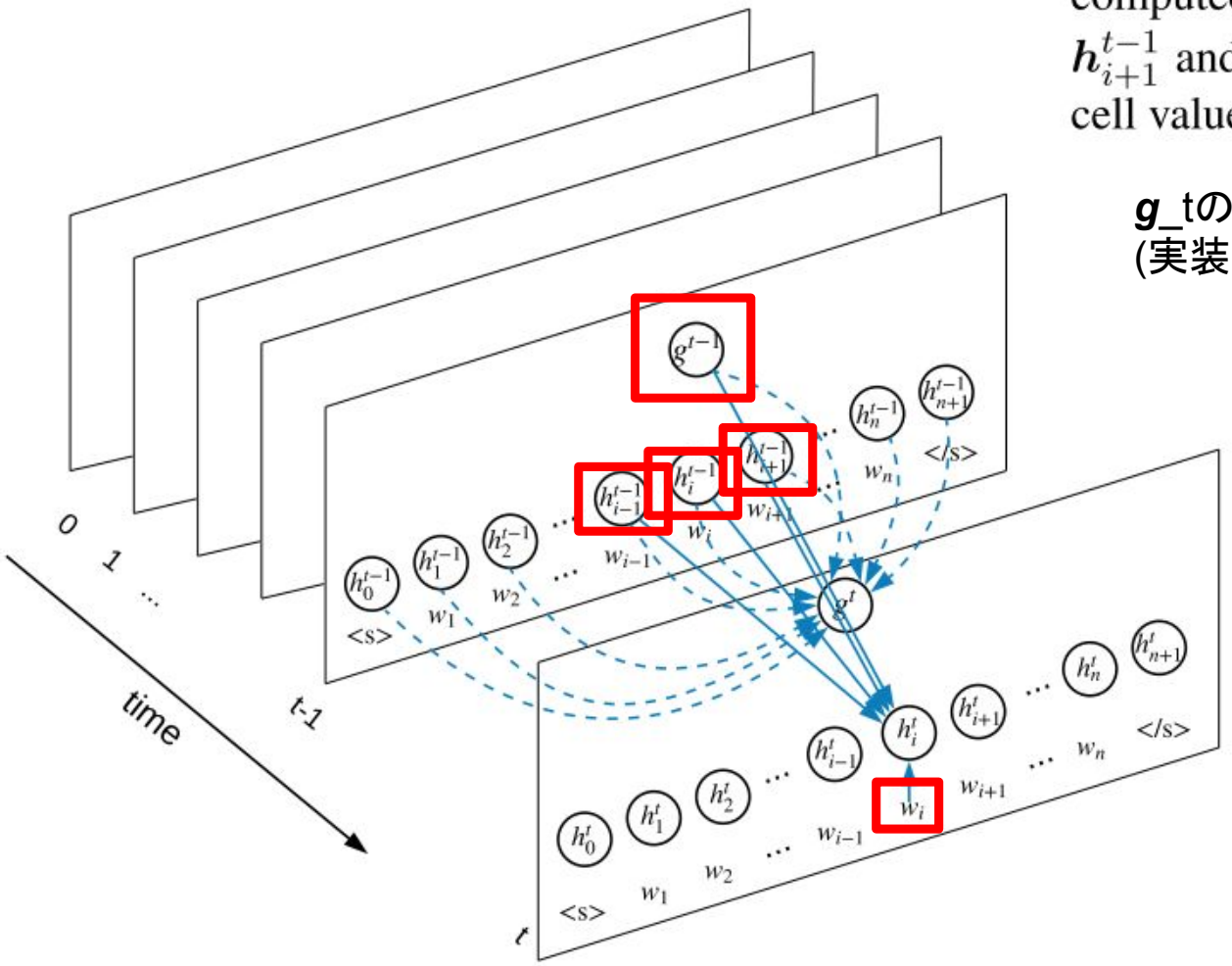
The BiLSTM model uses the concatenated value of $\overrightarrow{\boldsymbol{h}}^t$ and $\overleftarrow{\boldsymbol{h}}^t$ as the hidden vector for w_t:

$$\boldsymbol{h}^t = [\overrightarrow{\boldsymbol{h}}^t; \overleftarrow{\boldsymbol{h}}^t]$$

A single hidden vector representation $\boldsymbol{g}$ of the whole input sentence can be obtained using the final state values of the two LSTM components:

$$\boldsymbol{g} = [\overrightarrow{\boldsymbol{h}}^{n+1}; \overleftarrow{\boldsymbol{h}}^0]$$

Sentence-State LSTM for Text Representation



As shown in Figure 1, the value of each $\boldsymbol{h}_i^t$ is computed based on the values of $\boldsymbol{x}_i$, $\boldsymbol{h}_{i-1}^{t-1}$, $\boldsymbol{h}_i^{t-1}$, $\boldsymbol{h}_{i+1}^{t-1}$ and $\boldsymbol{g}^{t-1}$, together with their corresponding cell values:

g_tの生成過程まで式で述べられている
(実装見ないと多分全ては分からない)

Sentence-State LSTM for Text Representation

– 24 for test. For NER, we follow the standard split, and use the BIOES tagging scheme (Ratinov and Roth, 2009). Statistics of the four datasets are shown in Table 1.

NERにも使えるので、まずこれを目標にしたい！