Gapping parsing using pretrained embeddings, attention mechanism and NCRF

Emelyanov A. A., Artemova E. L.

MIPT/Sberbank, HSE

May 30, 2019

Plan

1. Walking on Sesame Street;
2. System architecture;
3. Training notes;
4. One word about errors.

The embeddings: BERT

1. Process input text sequence to WordPiece embeddings with a 30,000 token vocabulary and pad to 512 tokens.
2. Add first special BERT token marked “[CLS]“.
3. Mark all tokens as members of part “A“ of the input sequence.

The embeddings: weighting

Here we sum all of BERT hidden outputs:

$$o_i = \gamma \times \sum_{i=0}^{m-1} b_i s_i \tag{1}$$

where

- o_i is output vector of size 768;
- $m = 12$ is the number hidden layers in BERT;
- b_i is output from i BERT hidden layer;
- γ and s_i is trainable task specific parameters.

Recurrence

This part contains two LSTM networks for forward and backward passes with 512 hidden units so that the output representation dim is 1024 for each token.

Attention!

Apply Multi-Head Attention (from “Attention is all you need“) to the BiLSTM output. We took 3 heads and value and key dim 64.

Inference for sequence labelling

The representation form attention part is passed to Linear layer with *tanh* activation function and gets a vector with 14 dim, that equals to the to the number of entities labels (include supporting labels “pad“ and “[CLS]“). The inference layer takes the extracted token sequence representations as features and assigns labels to the token sequence. As the inference layer, we use Neural CRF++ layer. That extends the decoding algorithm with the support of *nbest* output.

Inference for classification task

For the classification inference, we use Pooling Linear Classifier block as proposed in ULMFiT paper. We pass output sequence representation H from Multihead-Attention part to different Poolings and concat:

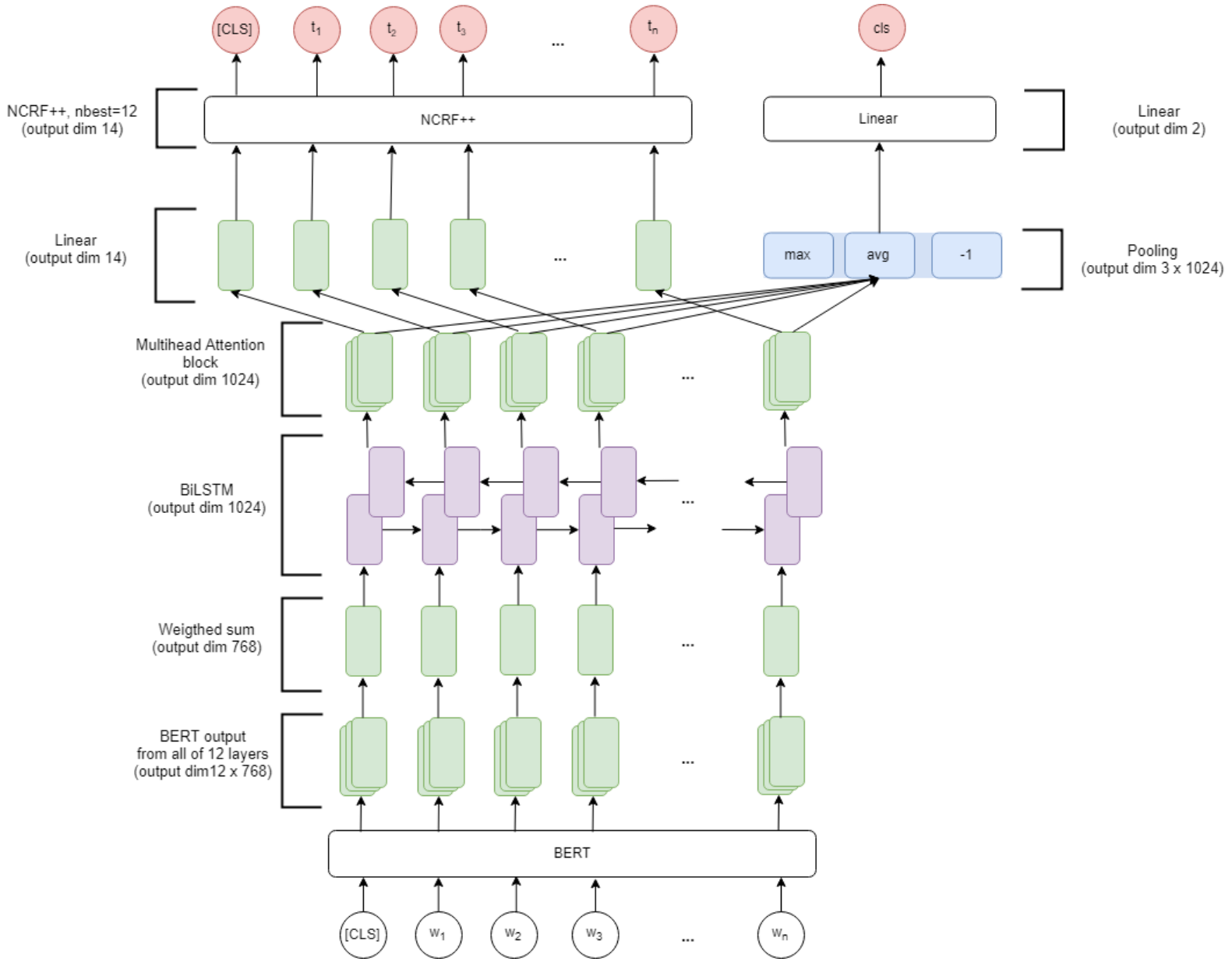
$$h_c = [h_T, maxpool(H), meanpool(H)] \tag{2}$$

where [] is concatenation;
h_T is last output significant vector of Multihead-Attention part (which does not have "pad" label).
The result of concat Pooling (3×1024) is passed to Linear layer, and that predicts binary classification.

Put everything together

1. BERT Embedder;
2. Weighted aggregation of BERT output;
3. Recurrent BiLSTM layer;
4. Multi-Head Attention;
5. Linear layer;
6. NCRF++ inference layer for sequence labelling;
7. Concat of differents Poolings for classification.

The system architecture



Post processing for the final result

1. Use classification result to weed out sequences without any labels.
2. If binary classification prediction is not 0 (gapping present) and predicted sequence labeling are returned.
3. Each WordPiece token in the word is matched with neural network label prediction. We use ensemble classifier on labels by count all predicted labels for one word except “X” and select label for a word with the higher number of votes.

Training Procedure

The proposed neural network was trained with joint loss:

$$\mathcal{L} = \mathcal{L}_{SL} + \mathcal{L}_{clf} \tag{3}$$

where $\mathcal{L}_{SL}$ is maximum log-likelihood loss for the sequence labeling task and $\mathcal{L}_{clf}$ is Binary Cross Entropy Loss for the classification task.

1. BERT-Adam (from Google);
2. Batch size - 16;
3. Epochs number - 100;
4. Around 5 GB of GPU memory.
5. Around five hours on one GPU.

Error analysis

1. Some errors with converting from origin data format (symbolwise markup) to word markup and back to origin after prediction. For example with extra spaces;
2. Bad Unicode symbols and there are some symbols, which are absent in WordPiece vocabulary;
3. Neural network prediction mistakes - network was overfitted on label “O“ and there are many false positives in prediction;
4. Bad learned structure of gapping.

Questions[1]

[1] Code is available at *https://github.com/king-menin/AGRR-2019*