

BERT

1. Use a **huggingface** library <https://github.com/huggingface/transformers> and follow their instructions to setup BERT FP32 model.
 - A. Then follow instructions at https://pytorch.org/tutorials/intermediate/dynamic_quantization_bert_tutorial.html to reproduce FP32 results as shown in web-site example (F1 = 0.9019).
2. Substitute **<bert.py>** in **../nlp_architect/models/transformers/** folder with the file provided in this package.
 - A. Comment/uncomment the appropriate part of the code to reproduce the specific task and method (REXP / 2D LUT) for selected precision (from source code line 310~). For example:
 - i. For **2D LUT** method precision **int16**
 - `attention_probs = softmax_LUT(attention_scores, exp_LUT_1x101_int16, Softmax_LUT_11x60_int16, 32768).float()`
 - ii. For **REXP** method
 - `attention_probs = softmax_LUT_rexp(attention_scores, -1).float()`
3. Put files **<icml2020_LUT.py>** and **<iv_rexp_LUT.py>** in the main folder of BERT
 - A. Comment/uncomment the appropriate part of the code to reproduce the **REXP** method for selected precision (from source code **<iv_rexp_LUT.py>** line 310~). For example:
 - i. **# int16 case (1x13 1x16)**
 - `rexp_LUT = torch.torch.ShortTensor(rexp_LUT_1x13_int16)`
 - `scale = 32768`
 - `expln_LUT = torch.torch.ShortTensor(expln_LUT_1x16_int16)`
 - `scale_eln = 32768`
 - ii. **# uint8 case (1x8 1x16)**
 - `rexp_LUT = torch.torch.ShortTensor(rexp_LUT_1x8_uint8)`
 - `scale = 256`
 - `expln_LUT = torch.torch.ShortTensor(expln_LUT_1x16_uint8)`
 - `scale_eln = 256`
4. Use **<bert_mrpc_quant.py>** or **<bert_sst2_quant.py>** to reproduce the results showed in our paper. For example:
 - A. To reproduce **BERT(MRPC)** results run
 - i. `python bert_mrpc_quant.py`
 - B. To reproduce **BERT(SST-2)** results run
 - i. `python bert_sst2_quant.py`