Here are some tips for you to build you method on our framework.

1. About the model

The name of our model is “Sp_norm_model”, and you can find it in “model.py”.

“self.gen” is encoder of the generator (a GRU layer). The input is a sequence with a shape [batch_size, sequence_len, word_embed_dim]. Then, it output the representation of the sentence: [batch_size, sequence_len, hidden_dim].

“self.gen_fc” gets an input of [batch_size, sequence_len, hidden_dim] and output [batch_size, sequence_len, 2]. Yes, we denote the mask of a token as a 2-dimensional vector. [batch_size, sequence_len, 2] is then fed into the gumbel-softmax layer (before a layernorm and a dropout). So, the mask of a token is [0,1] or [1,0]. And [0,1] means the token is selected as part of the rationale. So the rationale “z”=[batch_size, sequence_len, 2]

“self.cls” is encoder of the predictor (a GRU layer). For a normal predictor to do the classification, it gets an input with a shape of x=[batch_size, sequence_len, word_embed_dim]. Here we mask it with z: cls_embedding=x*z[:,:,1]. Then it’s trained in the same way as a normal classifier.

2. About the training steps

We first distinguish the parameters of the generator and the predictor in Line214-Line 223 of “norm_beer.py”. During training, we set different learning rates for the generator and the predictor according to the rationale sparsity, which is specified in Line48-Line54 of “train_util.py”.