AI504: Programming for AI (Fall 2020)

Final Project: Machine Translation

Due : Dec 16th, 2020, 11:59 pm

The goal of this project is improving the performance of Neural Machine Translation(NMT) system. In this project, you will tune the hyperparameters in NMT system without changing anything else (e.g. architecture, dataset, etc).

**0. Bugfix**

1. Please change the line in Data loader block like below.

|  |
| --- |
| !pip install --upgrade torchtext==0.8.0 |

1. **(Optional,** [**link to question in Classum**](https://classum.com/main/course/7726/149)**)** Please add the positional encoder to your model. You can add Type 1. positional encoding to your code. (Type 2. adds a learnable parameter to your model, so please do not use Type 2. positional encoder this time.)

< Type 1. sinusoidal positional encoding >

|  |
| --- |
| class Transformer(nn.Module):  def \_\_init\_\_(self, config):  super(Transformer,self).\_\_init\_\_()  self.encoder\_embedding = nn.Embedding(len(SRC.vocab),config.emb\_dim)  self.decoder\_embedding = nn.Embedding(len(TRG.vocab),config.emb\_dim)  self.pos\_encoder = PositionalEncoding(config.emb\_dim, config.dropout, max\_len = config.max\_position\_embeddings)  self.transformer = nn.Transformer(d\_model=config.emb\_dim, nhead=config.attention\_heads,  num\_encoder\_layers=config.encoder\_layers, num\_decoder\_layers=config.decoder\_layers,  dim\_feedforward=config.ffn\_dim, dropout=config.dropout, activation='gelu')  self.prediction\_head = nn.Linear(config.emb\_dim,len(TRG.vocab))    def forward(self, src, trg):  src\_token\_emb = self.encoder\_embedding(src)  src\_emb = self.pos\_encoder(src\_token\_emb)  trg\_emb = self.decoder\_embedding(trg)  output = self.transformer(src\_emb, trg\_emb,  tgt\_mask=self.transformer.generate\_square\_subsequent\_mask(trg.size(0)).to(device),  src\_key\_padding\_mask=src.eq(PAD\_IDX).permute(1,0).to(device),  memory\_key\_padding\_mask=src.eq(PAD\_IDX).permute(1,0).to(device),  tgt\_key\_padding\_mask=trg.eq(PAD\_IDX).permute(1,0).to(device))  prediction = self.prediction\_head(output)  return prediction  class PositionalEncoding(nn.Module):  def \_\_init\_\_(self, d\_model, dropout=0.1, max\_len=512):  super(PositionalEncoding, self).\_\_init\_\_()  self.dropout = nn.Dropout(p=dropout)  pe = torch.zeros(max\_len, d\_model)  position = torch.arange(0, max\_len, dtype=torch.float).unsqueeze(1)  div\_term = torch.exp(torch.arange(0, d\_model, 2).float() \* (-math.log(10000.0) / d\_model))  pe[:, 0::2] = torch.sin(position \* div\_term)  pe[:, 1::2] = torch.cos(position \* div\_term)  pe = pe.unsqueeze(0).transpose(0, 1)  self.register\_buffer('pe', pe)  def forward(self, x):  x = x + self.pe[:x.size(0), :]  return self.dropout(x) |

< Type 2. positional embedding >

|  |
| --- |
| class Transformer(nn.Module):  def \_\_init\_\_(self, config):  super(Transformer,self).\_\_init\_\_()  self.encoder\_embedding = nn.Embedding(len(SRC.vocab),config.emb\_dim)  self.decoder\_embedding = nn.Embedding(len(TRG.vocab),config.emb\_dim)  self.pos\_encoder = PositionalEncoding(config.emb\_dim, config.dropout, max\_len = config.max\_position\_embeddings)  self.transformer = nn.Transformer(d\_model=config.emb\_dim, nhead=config.attention\_heads,  num\_encoder\_layers=config.encoder\_layers, num\_decoder\_layers=config.decoder\_layers,  dim\_feedforward=config.ffn\_dim, dropout=config.dropout, activation='gelu')  self.prediction\_head = nn.Linear(config.emb\_dim,len(TRG.vocab))    def forward(self, src, trg):  src\_token\_emb = self.encoder\_embedding(src)  src\_emb = self.pos\_encoder(src\_token\_emb)  trg\_emb = self.decoder\_embedding(trg)  output = self.transformer(src\_emb, trg\_emb,  tgt\_mask=self.transformer.generate\_square\_subsequent\_mask(trg.size(0)).to(device),  src\_key\_padding\_mask=src.eq(PAD\_IDX).permute(1,0).to(device),  memory\_key\_padding\_mask=src.eq(PAD\_IDX).permute(1,0).to(device),  tgt\_key\_padding\_mask=trg.eq(PAD\_IDX).permute(1,0).to(device))  prediction = self.prediction\_head(output)  return prediction  class PositionalEncoding(nn.Module):  def \_\_init\_\_(self, d\_model, dropout=0.1, max\_len=512):  super(PositionalEncoding, self).\_\_init\_\_()  self.dropout = nn.Dropout(p=dropout)  self.pe = nn.Embedding(max\_len, d\_model)  def forward(self, x):  position = torch.arange(0, x.size(0)).to(self.pe.weight.device)  x = x + self.pe(position).unsqueeze(1)  return self.dropout(x) |

**1. Problem**

The dataset and task of this project is same as Practice week 11. Below is the description of Multi30k translation dataset.

|  |
| --- |
| As a translation task, this task consists in translating English sentences that describe an image into German, given the English sentence itself. As training and development data, they provide 29,000 and 1,014 triples respectively, each containing an English source sentence, its German human translation. As test data, we provide a new set of 1,000 tuples containing an English description. |

**2. To do**

You should modify the first block of the template code. Below is the meaning of each parameter.

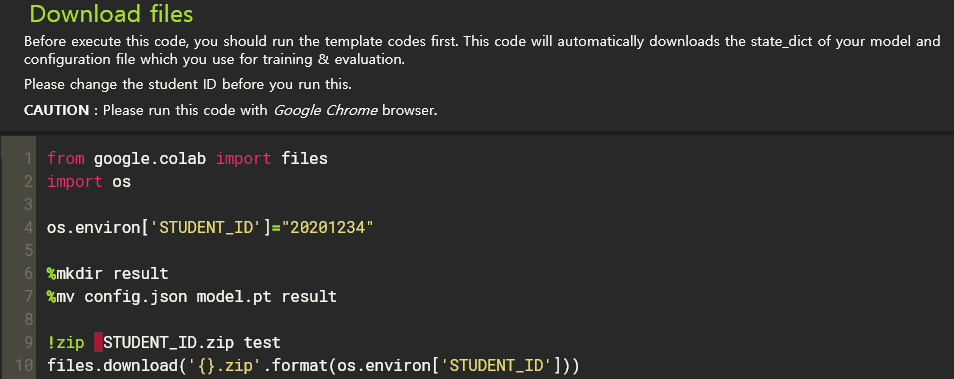
|  |
| --- |
| * emb\_dim : hidden dimension of input embedding * ffn\_dim : hidden dimension of position-wise feed forward network * attention\_heads : number of attention heads * dropout : dropout rate * encoder\_layers : number of encoder layers * decoder\_layers : number of decoder layers * lr : learning rate * batch\_size : size of batch * nepochs : number of epochs to train |

**The threshold of BLEU score which you need to exceed is as follows.**

* If you pass the midterm project : 0.20
* If you fail the midterm project : 0.22
* If you did not submit the midterm project : 0.26

**3. Submission guide**

After you finish the training, run the second block of the template code to download files.

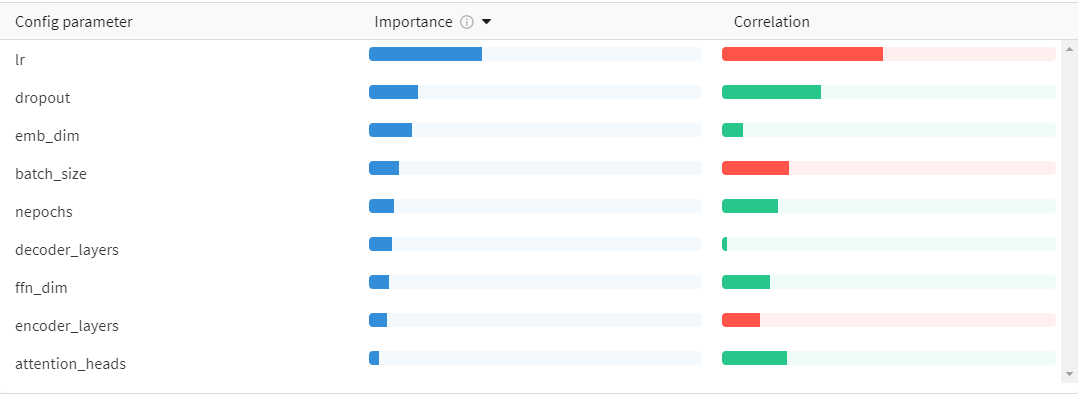


It’ll compress the state dict of the trained model and your hyperparameter configurations into a <student\_ID.zip> file. If you run this code on Chrome browser, then it’ll automatically download the compressed zip file to your local machine. Please submit the downloaded .zip file to KLMS.

**Caution : Change the sample student ID(20201234) to your student ID.**

**4. Hint : importance of hyperparameters**

TAs have done some hyperparameter search and measure the importance of hyperparameters. Higher importance means that hyperparameter has more impact on BLEU score. We hope it helps your hyperparameter tuning.

****

Q. What is the meaning of an importance and correlation?

A. **Correlation** is the linear correlation between the hyperparameter and the chosen metric (in this case BLEU score). So a high correlation means that when the hyperparameter has a higher value, the metric also has higher values and vice versa.

**Importance** shows you the degree to which each hyperparameter was useful in predicting the chosen metric.