llm---bert

encoder로 동의어관계 파악/추출, decoder는 생성형이 아니므로 필수적이지 않음

만드는중

1) 데이터셋 : hugging face의 ("glue", "mrpc") , sentence_1 , sentence_2 사이의 동의어 관계 분석을 위한 구조를 가지고 있음

['sentence1', 'sentence2', 'label', 'idx']

2) 데이터시각화

i) label distribution : Not equivalent(=0)이 equivalent(=1)에 비해 대략 절반 정도다.-> calss mismatch이므로 bce_loss에서 equivalent label의 경우, pos_weight을 0.5로 조정해 loss 수식에서의 영향력을 절반 정도로 줄인다

ii) sentence length distribution : 가지각색이다. zero padding은 64로 맞추면 되지만, sentence length의 frequency가 동일하지 않고 대략 sentence length 20~25를 중심으로 몰려있다. 어떤 데이터는 주변부 등 자잘한 의미를 더 많이 담고 있을 가능성이 높다.

3) 토큰화. pretrained bert model로 tokenizer를 생성해 문장을 토큰화한다.

4) hyperparameter에서 embedding_dim = 64(=sentence_length) * 8(=num_heads)이다.

5) pre_process

```
class Pre_process(nn.Module):
    def __init__(self ,num_heads , embedding_dim , vocab_size = 40000 , max_length = 64, dropout=0.1):
        super().__init__()
        self.embed_size = embedding_dim
        self.max_length = max_length
        self.word_embedding = nn.Embedding(vocab_size, embedding_dim)
        self.position_embedding = nn.Parameter(torch.randn(1 , max_length , embedding_dim))
        self.dropout = nn.Dropout(dropout)
    def run(self , input , token_type_ids , attention_mask):
        batch_num , seq_length = input.shape
        z = self.word_embedding(input) # z.shape : batch_num , sequence_length , embedding_dim
        positional_embed = self.position_embedding.expand(batch_num , self.max_length , self.embed_size)
        # 단어 임베딩 + position + 문장 소속(binary)
        out = z + positional_embed + token_type_ids.unsqueeze(2).expand(batch_num,self.max_length,self.embed_size) + attention_mask.unsqueeze(2).expand(batch_num,self.max_length,self.embed_size)
        out = self.dropout(out)
        return out # out.shape : batch_num , sequence_length , embed_size
```