Final Project: Machine Learning for Natural Language Processing

Xianlin DING

March 22, 2023

Github : https://github.com/savourdxl/NLP-ENSAE-3A-Topic3

1 Introduction

Our project is focused on intent classification (topic 3). Basically, we want to use transformer to solve sequence labelling problem.

The data set used for classification is SILICONE, which includes several labelling tasks with both DA and E/S annotations. Specifically, the data is in multiple levels: we have a set of conversations $D = (C_1, C_2, \ldots, C_{|D|})$, where each conversations is composed of utterance $C_i = (u_1, u_2, \ldots, u_{|C_i|})$ and each utterance is a sequence of words. The following table gives us an example of one conversation with DA labels on utterances.

Table 1: Example of conversation with labelled utterances

Utterances	Dialogue Act(DA)
"what do you mean ? it will help us to relax ."	"question"
"good.let ' s go now ."	"directive"
"all right ."	"commissive"

2 Model Description

The basic structure of our model can be described as follows, where basically we first use transformer decoder layers to transform data, then we pass them to the MLP classifier to get the results.

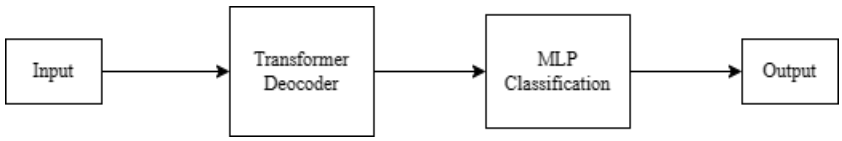

Figure 1: Structure of Model

2.1 Transformer Decoder

The process of using transformer decoder can be divided into 4 steps.

Firstly, we tokenize each utterance. Notice that each utterance is composed of a sequence of words $u_i = (w_{i1}, w_{i2}, ..., w_{ik})$, but the length of words in each utterance is different. In this case, we set the length of words is equal to 20: for the sequence with length more than 20, we delete the words that exceed the length; for the sequence with length less than 20, we fill the blanck with 1. Here, we use FastText (English) from torchtext.vocab.

Secondly, we generate the embedding h_0 on $U = (u_1, u_2, \ldots, u_n)$: $h_0 = U^W * W_e + W_p$, where W_e is the token embedding matrix, W_p is the position embedding matrix and w_i stands for the words.

Thirdly, we transformer the output by using 12 transformer decoder blocks. In mathematical terms, it can be written as $h_l = transformer\_block(h_{l-1}), l = 1, \ldots, 12$. In general, this pre-training can be viewed as the maximization of the function $L_1(C) = \sum_i \log P(u_i|u_{i-k}, \ldots, u_{i-1}; \Theta)$, where Θ refers to the parameters of the language model, and k refers to the length of window.

However, there is a need to notice that the transformer block here is not the usual encoder-decoder one ([2]), but the transformer decoder: masked multi-head self attention + multi-head self attention (the number of heads in our model is equal to 10) + feed forward. In other words, it is a model with single direction (left-to-right transformer), i.e., we can only infer one word from the past words. The Figure 2 shows the structure of this transformer.

In fact, the masked multi-head self-attention is the key for our transformer. The attention of each head without mask is computed as $Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V$ ([1]), where Q is the query, K is the key, V is the value and d_k is the key dimension. Hence, the attention of each head with mask is computed as $Masked\_Attention(Q, K, V, M) = softmax(\frac{QK^T+M}{\sqrt{d_k}})V$, where M stands for the mask and anything else is the same.

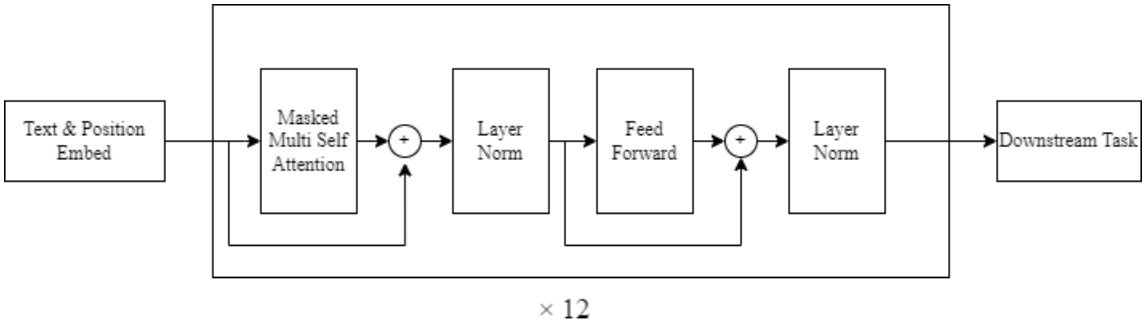

Figure 2: Structure of Transformer Decoder

Finally, we pass the outputs of pre-training for the softmax function with linear combination. This step can be viewed as the maximization of the function $L_2(C) = \sum \log P(Y|x_1, \ldots, x_m)$, where $P(y|x_1, \ldots, x_m) = softmax(h_{12}W_y)$ and h_{12} is the output after 12 transformer decoders.

In this case, the current total function that we want to maximize is $L_3(C) = L_2(C) + \lambda L_1(C)$.

2.2 Classifier : Multi-layer Perceptron

Now we can use classification model. The classifier we used is a 3-layer MLP, where neurons in the last layer refer to all intent classes. In this case, we adopted cross entropy loss as loss function, whose equation can be written as follows, where n stands for the number of classes, y_{ji} stands for the actual value of the probability of class i and $\hat{y_{ji}}$ stands for the real value. In fact, the cross entropy measures the degree of different information between two distributions.

$$Loss = \frac{1}{batch\_size} \sum_{j=1}^{batch\_size} \sum_{i=1}^{n} -y_{ji} \log y_{ji} - (1 - \hat{y_{ji}}) log(1 - \hat{y}_j)$$

The Figure 3 shows the basic structure of our Multi-layer Perceptron, where the first layer is the input layer (the utterance from the output of pre-training model), the last layer is the output layer (the probability of one class), and the second layer is the hidden layer (we have two hidden layers in our case). During our training, we do the random initialization of weights in the MLP to avoid that the coefficients of fitting function don't change at all during the training. The activation function is ReLU() function.

To sum up, the structure of model's layer can be summarized as follows:.(1) Transformer decoder layers: embedding layer, transformer decoder layer*12, linear layer, linear layer, SoftMax layer; (2) Classifier Layers: linear layer, ReLU layer, linear layer, ReLU Layer, linear layer, SoftMax layer.

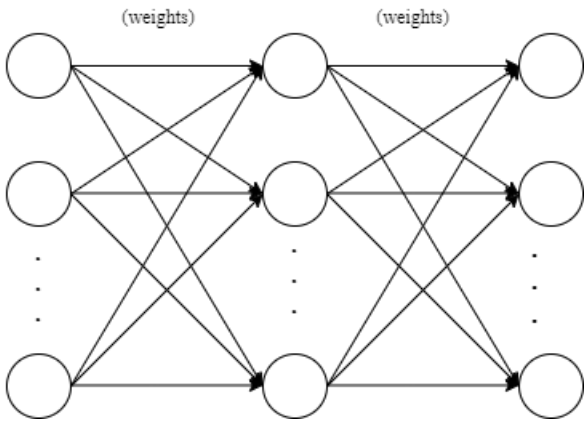

Figure 3: Structure of Multi-layer Perceptron

3 Results

Due to time limit, we only consider the subset 'dyda_da' of SILICONE for our training example. Besides, as a simple example, we only collect the first 1000 training data in this subset to construct all of our data.

As usual, we split the collected data set into training set (670 observations), validation set (165 observations) and testing set (165 observations). In this data set, the utterances are with four types of labels: 0 (commissive), 1 (directive), 2 (inform), 3 (question). After data checking, we find that the data set in terms of label is not balanced. In this case, we consider weighted cross entropy loss when do the optimization of the model. The optimizer we use is Adam algorithm.

Figure 4 show us if the training process goes well. We can see that in general, the weighted cross entropy loss decreases when we run the epoch one by one (in our examples, we have ran 5 epochs).

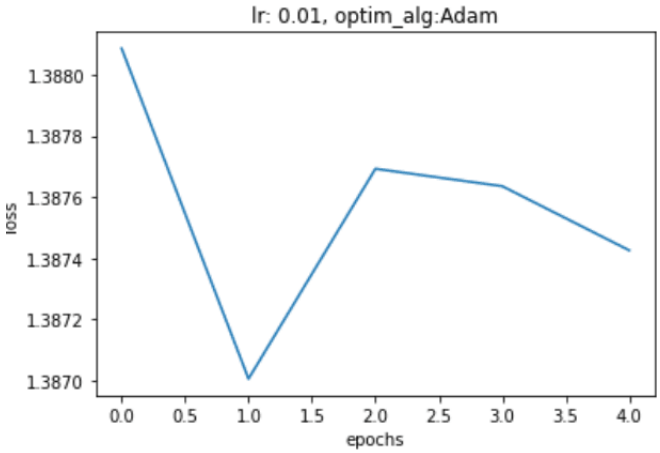

Figure 4: Loss with epoch running

Figure 5 instead shows us the situation of accuracy rate when we run the epoch one by one. As we can see in the picture, the accuracy rate doesn't not always increase, but is under fluctuation. The value on the whole is around 52% .

When we test the training model on the testing data, the weighted cross-entropy loss is equal to 1.3822, and the accuracy rate is 55.47%. And we also notice that the classification model is not good enough, since our training model tends to classify the the utterances into 2 of the 4 categories (label 2 and label 3). The weighted cross-entropy loss on validation data is 1.3817, and the corresponding accuracy rate is 60.16%.

Several reasons cause for this. On one hand, we only run very few of the data (subset 'dyda_da'

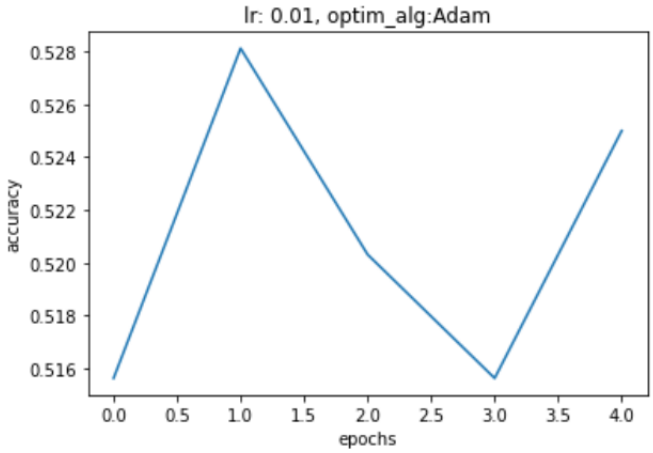

Figure 5: Accuracy rate with epoch running

has around 80000 training data). On the other hand, the transformer layers here may need to be ameliorated and we may consider introducing pooling layers or convolutional layers to extract more prominent features in the tensors.

Other tables can be summarized using the same code in our Github. For instance, we can change the learning rate (in this example, learning rate is equal to 0.01), optimization (SDG for example) , loss function (NLL loss for example)and percentage of training set split (67% in this example) to see how the loss and accuracy rate will change. We need also to train this model on other subsets in SILICONE. Due to time limit, the content here is passed.

Finally, for our training model, we have 760310743 parameters in the whole training structure.

References

[1] Lucas Chaves Lima Christian Hansen Maria Maistro Jakob Grue Simonsen Dongsheng Wang, Casper Hansen and Christina Lioma. Multi-head self-attention with role-guided masks. *European Conference on Information Retrieval*, LNISA, volume 12657, 2021.

[2] Matteo Manica Matthieu Labeau Chloe Clavel Emile Chapuis, Pierre Colombo. Hierarchical pre-training for sepquence labelling in spoken dialog. *Findings of the Association of Computational Linguistics*, EMNLP 2020, 2021.