Automatically verify keyword-description relationship

Goal:

1. when a shop buys a keyword from Rakuten, we can check if the keyword is relevant to the item he sells.

There are two approaches:

1. Using keyword extraction models (reference: Model - Keyword extraction)
2. Using two encoders to encode the keyword and description, respectively. Then make a prediction. (the following model)

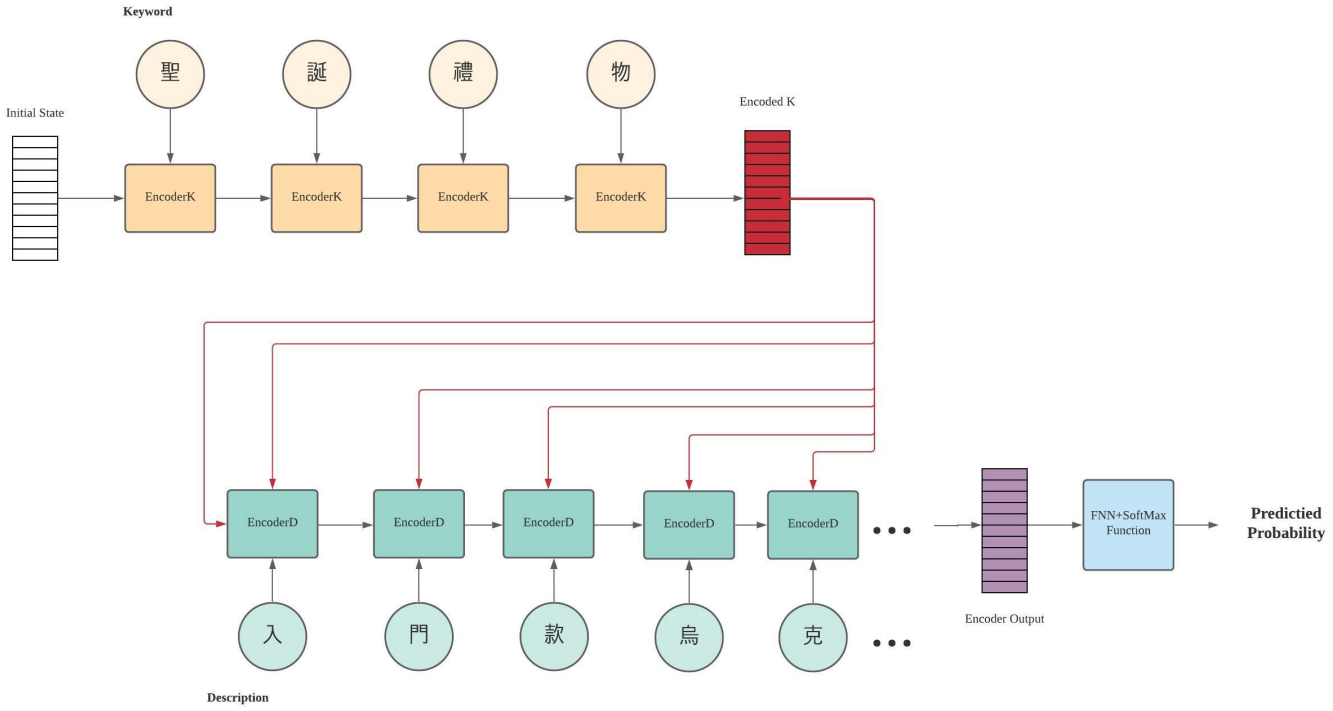
Concepts:

The idea is that: we want our model to mimic how we evaluate the relation, which is: first we remember the keyword and we see through the sentence to see if there is the pattern of our keyword in the sentence.

The training steps:

1. We feed our keyword into the EncoderK to encode the keyword to a vector.
2. We concatenate the encoded keyword to every word of the sentence to make sure our model remember the keyword when reading the sentence (one can also use the Attention mechanism, but since our keywords are usually very short here, I don't think that is necessary)
3. We feed the concerted sentence to the EncoderD to make predictions.

Example:



Training data:

We use the customer-searched result as training data. If a customer searched a keyword and then clicks on a product, we bind the searched keyword and the description of the clicked product together. This is the training data that our targets, y=1. We used 12.5 million rows of such data.

Then, to produce data that have target = 0, we duplicate the above data and randomly assigns keyword to descriptions.

In this way, we produced 25 million rows of totally balanced data. Nothing can be used to predict except the link of keyword and description.

Result:

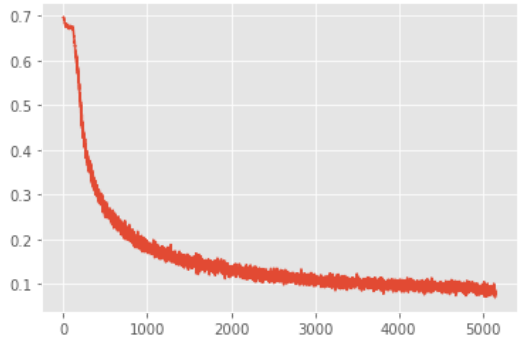
After training for about 10 days on GPU(V100). We achieved 97.15% accuracy in the testing data the model has not seen. And more than 94% accuracy in the real-life data comparing with the one labeled by hand before.

```
In [56]: avg_loss, right_num, total_num = evaluate(train_loader,encoderK1,encoderD1,threshold = 0.5,sample_size = 50000,print_index=False)
         print("right: ",right_num,"  out of: ",total_num,"  Accuracy: ",right_num/total_num)
         print("avg loss: ",avg_loss)
```

```
100%|██████████| 49995/50000 [02:37<00:00, 305.00it/s]

right:  48578   out of:  50000   Accuracy:  0.97156
avg loss:  0.08254057010124276
```

loss after one epoch



Sample code:

Note: All the code here is re-contructed by me out of work-time and is not the one implemented in the company.