**Assignment 4 Natural Language Inference**

**Tean #4 Wenjun song**

**Haiming zhang**

**Yanjun lyu**

Natural language inference is the task of determining whether a “hypothesis” is true (entailment), false (contradiction), or undetermined (neutral) given a “premise”. Reasoning and inference are central to both human and artificial intelligence.Natural languageinference (NLI), also known as recognizing tex-tual entailment (RTE), is an important NLP problem concerned with determining inferential relationship (entailment, contradiction, or neutral) between premise and hypothesis. In general, modeling informal inference in languageis a very challenging and basic problem towards achieving true natural language understanding.

**1. Dataset--SNLI**

The Stanford Natural Language Inference (SNLI) Corpus contains around 550k hypothesis/premise pairs. Models are evaluated based on accuracy.

**2. Reference paper**

2.1. Enhanced LSTM for Natural Language Inference, written by Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, Diana Inkpen.

2.2. Learning Natural Language Inference with LSTM, written by Shuohang Wang, Jing Jiang.

2.3. An overview of Natural Language Inference Data Collection, written by Stergios Chatzikyriakidis, Robin Cooper, Simon Dobnik, Staffan Larsson.

2.4. Supervised Learning of Universal Sentence Representations from Natural Language Inference Data, witten by Alexis Conneau, Douwe Kiela, Holger Schwenk, Loic Barrault, Antoine Bordes

**3. Understanding**

3.1 From SNLI websits, there are many models refering natural language reference using SNLI dataset. We research some of them to build our own model. Figure(1) is for the models which are from different thinking dimentions.

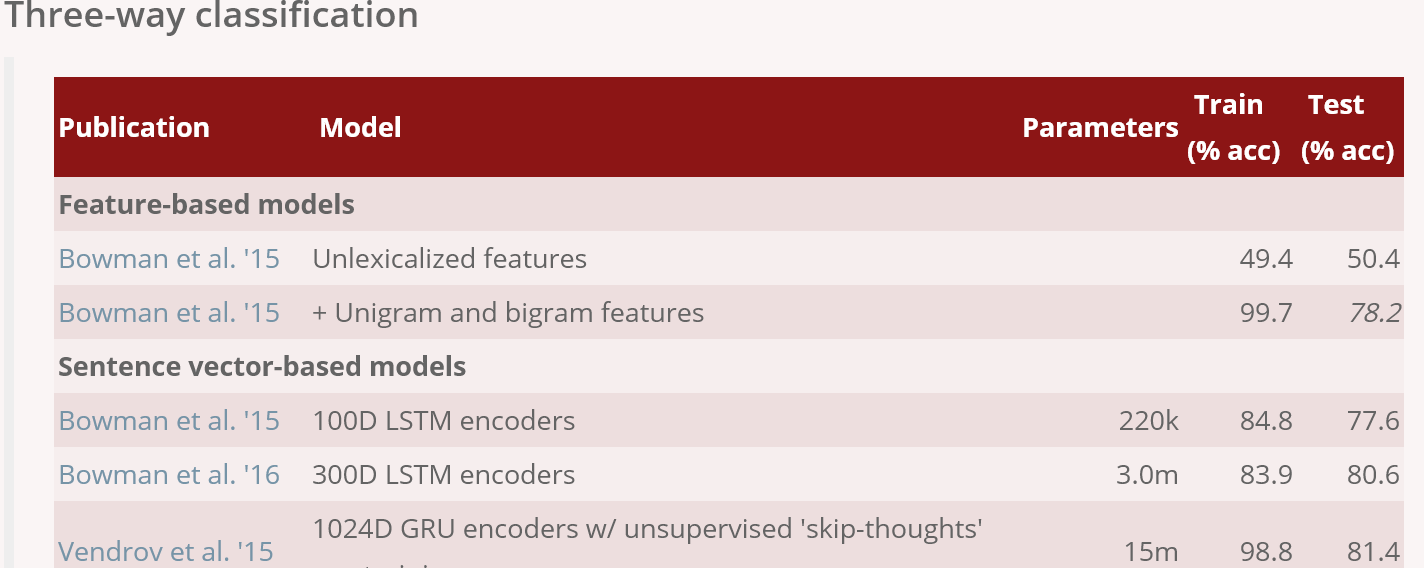


Figure 1

3.2 There are three types of NLI models: Feature-based models, Sentence encoding-based models, Other neural network models. Our research targets mainly focus on Sentence encoding-based models which is the main domain of NLI.

LSTM encoders:

We can use h and p to encoder the input data, and then put the outcome to three fully connections layers with activation function, next we use a softmax function to output the final. This model is simple for NLI, and there are many papers were based on this model, such as Tree-based CNN encoders, and InferSent, Structured Self Attention, mLSTM & word-by-word attention, Decomposable Attention, enhanced sequential infer-ence model (ESIM), and KIM & DMAN.

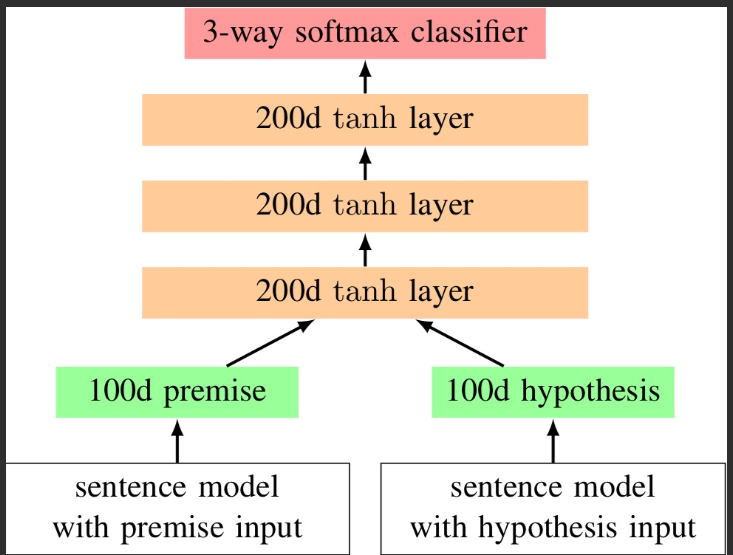
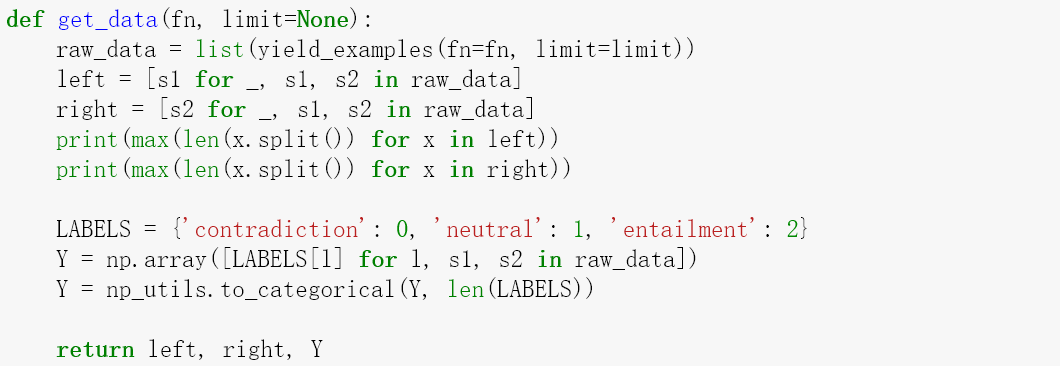


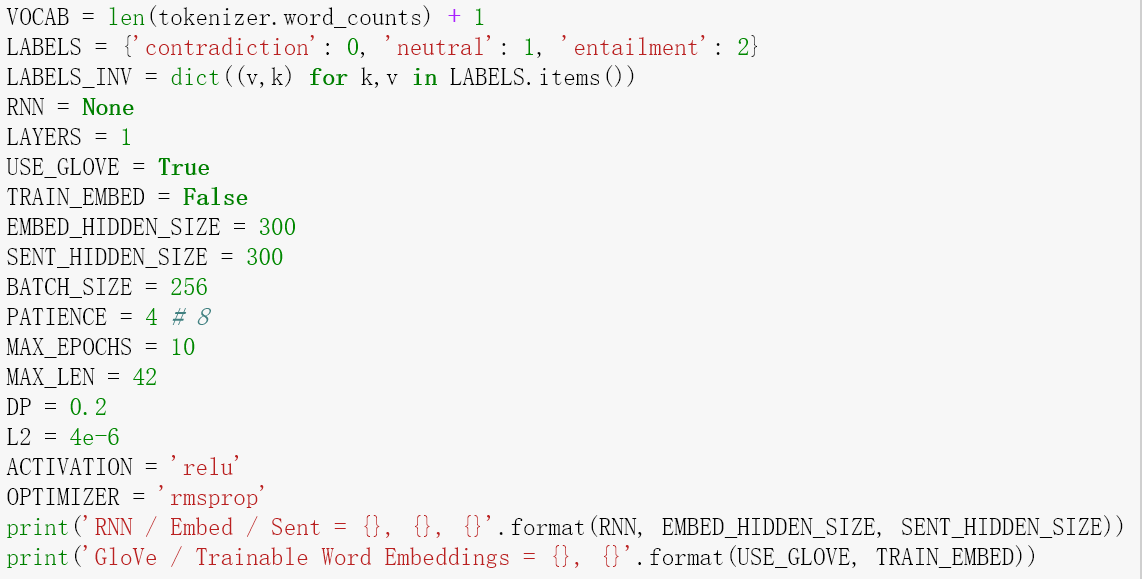
Figure 2 LSTM encoders

We use this model to train the SNLI dataset to analyze the relationship between premise and hypothesis.

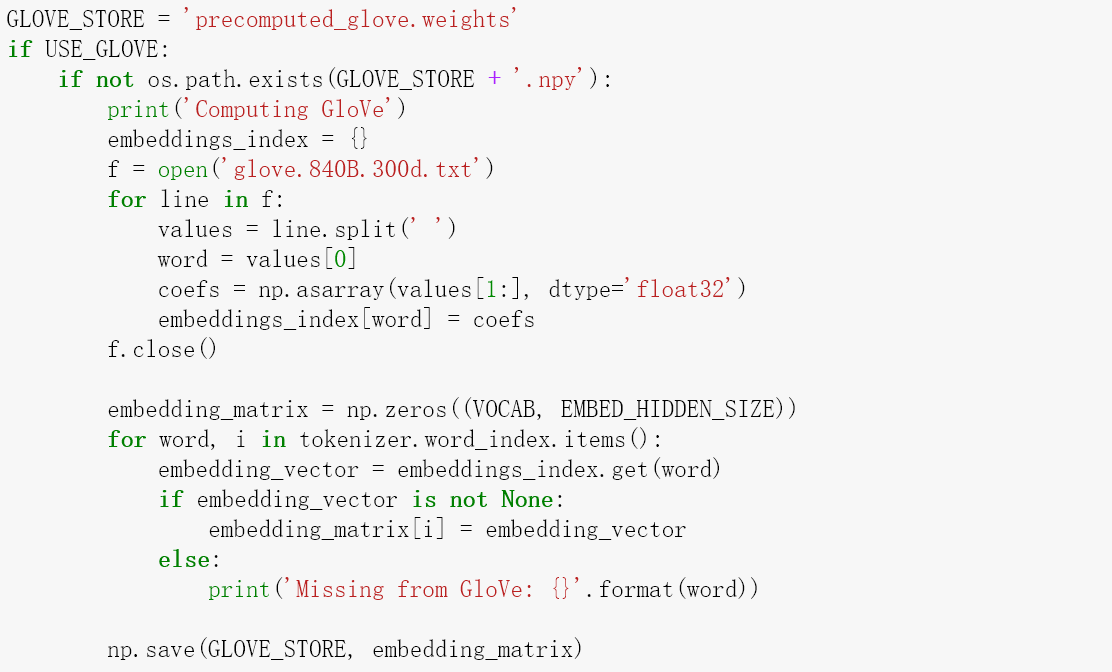
Firstly, we should split the dataset into three parts: premise and hypothesis and the labels.



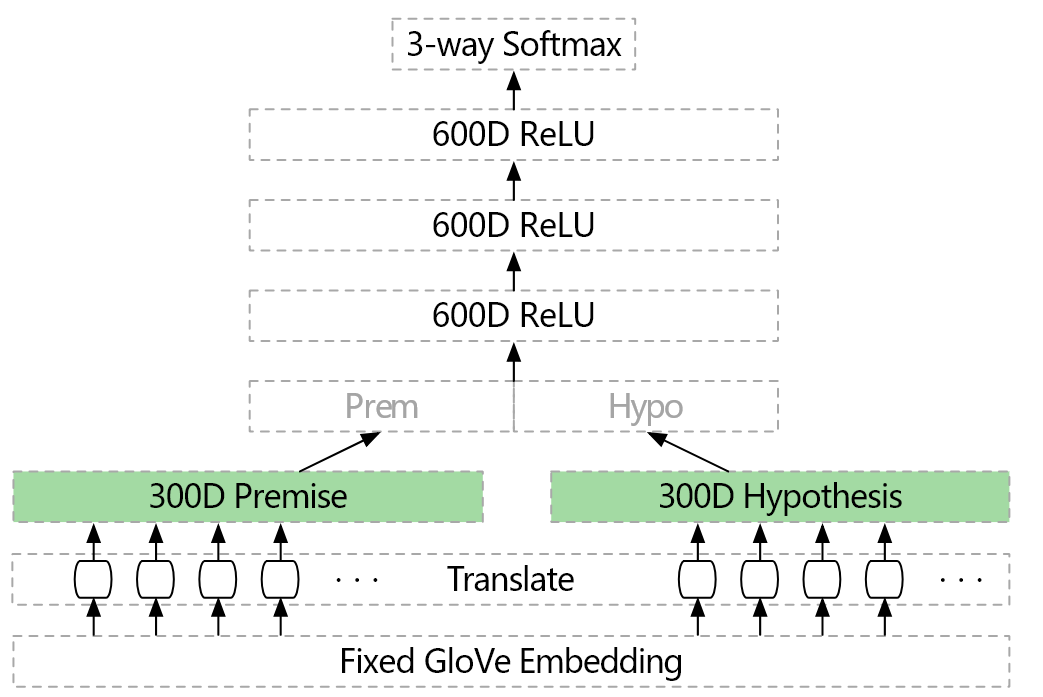
Then we tokenizing the data, and set the paremeters of the encoding model:



Then we download the pre-trained model-- glove.840B.300d.txt from GLOVE model website to train SNLI dataset like this:



The model architecture is like this:



Extract a 300D word vector from the fixed GloVe vocabulary

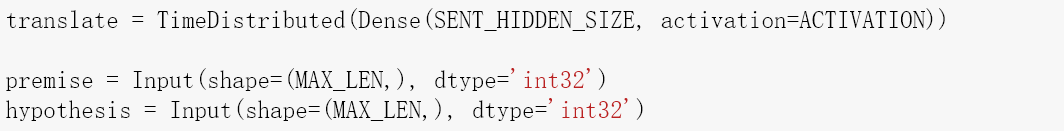
Pass the 300D word vector through a ReLU "translation" layer

Encode the premise and hypothesis sentences using the same encoder (summation, GRU, LSTM, ...)

Concatenate the two 300D resulting sentence embeddings

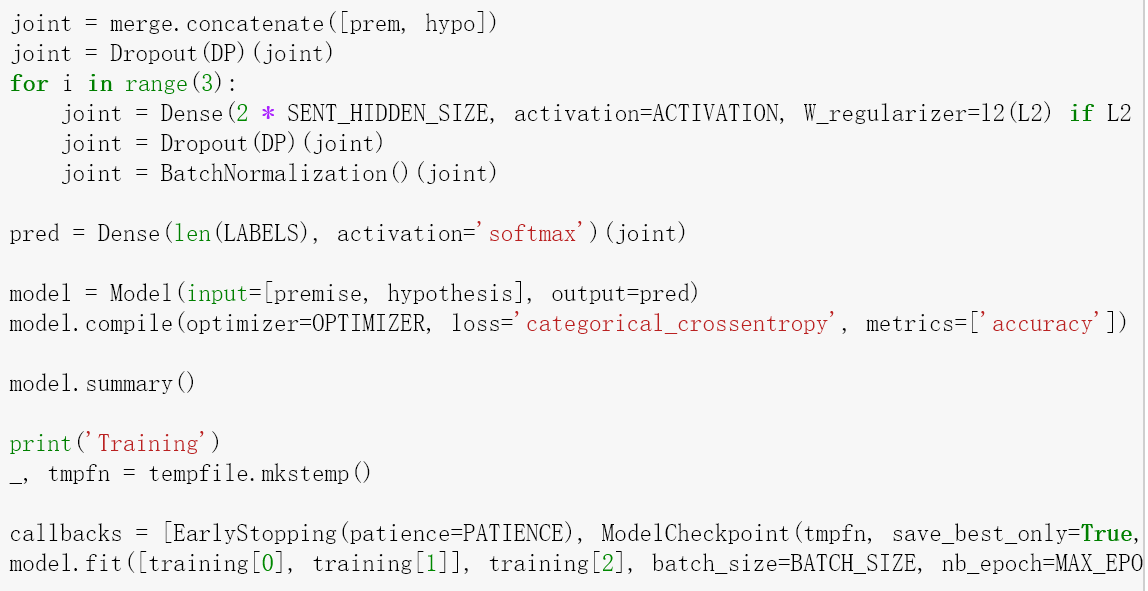
3 layers of 600D ReLU layers

3 way softmax



Due to the output is three catogorical values—“contradiction”, “neutral”, “entailment”, we just need to get the accuracy for test data to metric our model performance.

So this is the final model details.



\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param # Connected to

==================================================================================================

input\_9 (InputLayer) (None, 42) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

input\_10 (InputLayer) (None, 42) 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

embedding\_5 (Embedding) (None, 42, 300) 12717300 input\_9[0][0]

input\_10[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

time\_distributed\_5 (TimeDistrib (None, 42, 300) 90300 embedding\_5[0][0]

embedding\_5[1][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

lambda\_5 (Lambda) (None, 300) 0 time\_distributed\_5[0][0]

time\_distributed\_5[1][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_18 (BatchNo (None, 300) 1200 lambda\_5[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_19 (BatchNo (None, 300) 1200 lambda\_5[1][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

concatenate\_5 (Concatenate) (None, 600) 0 batch\_normalization\_18[0][0]

batch\_normalization\_19[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_13 (Dropout) (None, 600) 0 concatenate\_5[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_18 (Dense) (None, 600) 360600 dropout\_13[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_14 (Dropout) (None, 600) 0 dense\_18[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_20 (BatchNo (None, 600) 2400 dropout\_14[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_19 (Dense) (None, 600) 360600 batch\_normalization\_20[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_15 (Dropout) (None, 600) 0 dense\_19[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_21 (BatchNo (None, 600) 2400 dropout\_15[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_20 (Dense) (None, 600) 360600 batch\_normalization\_21[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_16 (Dropout) (None, 600) 0 dense\_20[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

batch\_normalization\_22 (BatchNo (None, 600) 2400 dropout\_16[0][0]

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

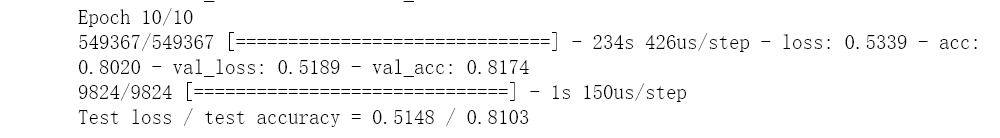
dense\_21 (Dense) (None, 3) 1803 batch\_normalization\_22[0][0]

==================================================================================================

Total params: 13,900,803

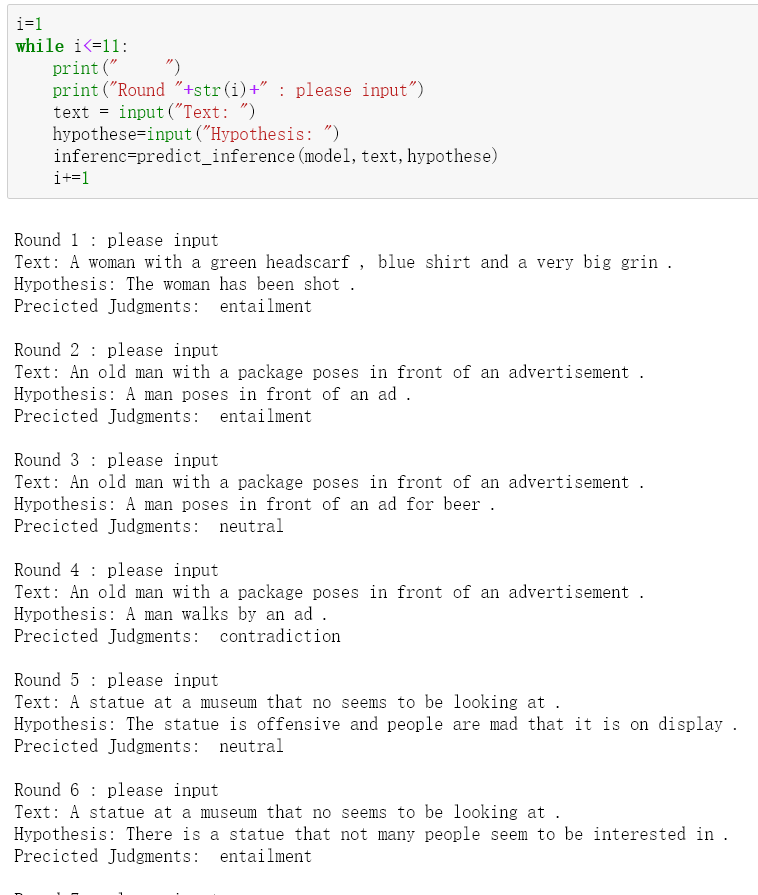
Trainable params: 1,178,703

Non-trainable params: 12,722,100

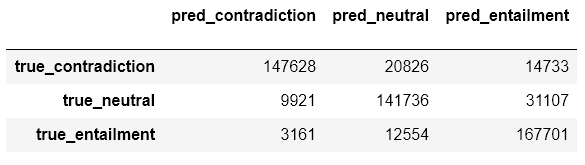


Basically this is a good model for SNLI dataset, the test accuracy is 0.81 which is enough for inferencing sentences for the realistic problems. However, it is certainly not the best model, we should make more research to develop and evolve the model to get a better output.

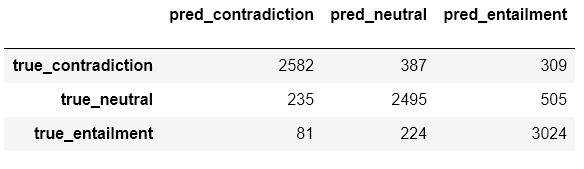
Then we put 11 pairs of sentences to do the “unit test and show the results” like this:



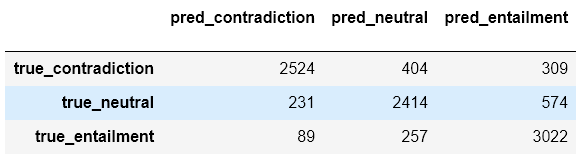
Then we made three confusion matrixes to help us to observe this model for the training data, validation data and test data.



The accuracy of training data is 0.83



The validation accuracy is 0.82



The test accuracy is 0.81

From the outcome above, we can see the SNLI dataset is a mature one to train a NLI model, and the model we used performanced well for this data. The accuracy for training, validation, and test is around 0.81-0.83 which can inference a relative better output for the real problems. Maybe we can find a better model to have a better output if we make a deeper researchwith more data or with other datasets.