Natural Language Processing

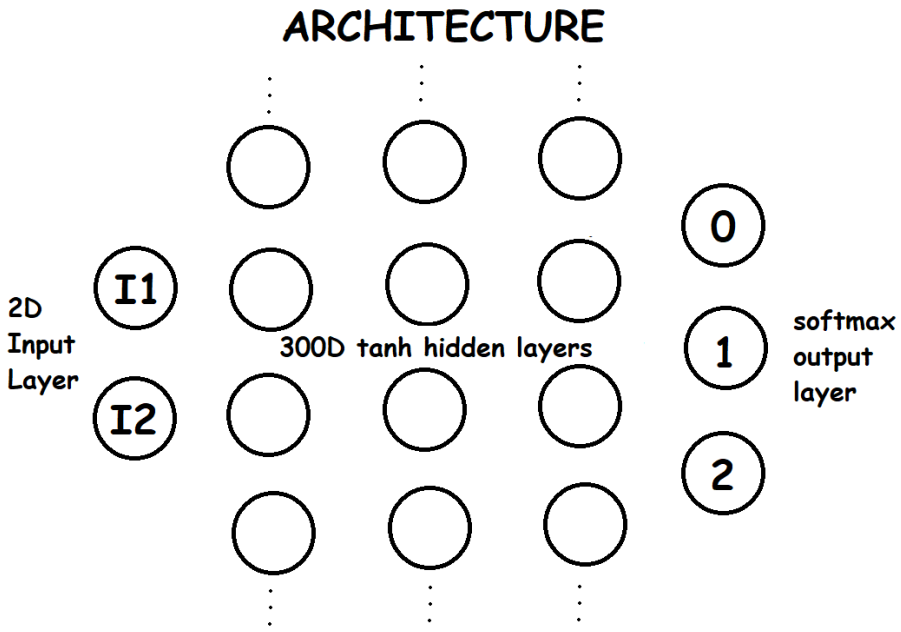
This question is a combination of two tasks. (a) Natural language Inference and (b) Probing the sentence embeddings to interpret what it is learning.

The task begins by formatting the data according to our convenience. Firstly, I unzipped the SNLI dataset then merged all the files into a single text file named “SNLI.txt”. I didn’t write a script for this, it was simple ctrl+a, ctrl+c, ctrl+v. Next, I wrote the files, ”1 Cleaning.py” and “2 Some Preprocessing.py”. (Note: You must run all the scripts in the given order, only once, except for the first script. Follow the instructions given in the script.) They create the files “Raw_Data.txt” and “Word2Vec.txt” which are just readable versions of the “SNLI.txt” file.

Then I used the “Word2Vec” model from “gensim” to create my own word embeddings. I decided not to use “GloVe” (global vectors for word representation) as it did not have some of the words that were in the dataset. I decided to create a 300-dimensional word vector representation for each word in the dataset. This is present in the file, “3 Word Embedding Creator.py” and after running it, 3 files starting with “True_Word_Embeddings” will be generated.

Script “4 Sentence Embeddings and Sent_Lens.py” actually creates the word embeddings and also the sentence encoding that we will be using. It also creates a file called, “Sent_Len.npy” which will be used later for probing.
For the sentence encoding, I decided to just take the average of the individual word vectors instead of using a more sophisticated method like LSTM, etc. because it was considerably faster and easier to understand.

Next is “5 Model.py” where we split the dataset into training and testing sentences and make the actual model. I have used a very simple model, which uses the below architecture.



Here we take the encoded sentence version of premise and hypothesis as inputs, concatenate them, and then put them through 3 dense layers with tanh activation and a 3-way softmax output layer.

The other parameters are given in the script but I only have a vague idea about what the “L2 regularization”, “adam optimizer” and the “categorical_crossentropy loss function” do.
After 10 epochs the model gives about a 69% accuracy on the testing data and after 30 it’s around 71. So if you want to save some time, change the epochs to 10.
Now for the probing assignment, run the file named, “6 Probing Sent_Len.py”. Although the question asks to use POS tagging for the auxiliary task I decided to investigate whether the **length of the sentence** is being encoded. Note: The length of a sentence is defined as the number of words in the original sentence.
After some investigation, I realized that there are almost 50 different values for the sentence length ranging between 1-81. So I decided to encode it as a 10-way classification problem where it must identify the length of the sentence in intervals of 10. (example: if sentence length is 21 words, then it must choose the 20-30 bracket.) Surprisingly it is very good at this task giving an accuracy of around 85% (This can either be because over 80% of the sentences are in the first two groups, or the property is actually getting encoded. Either way further testing is needed.)
The actual model for the probe is just an input layer, a 300-dimensional hidden layer with rectified linear activation function, and a 10-way softmax output layer.