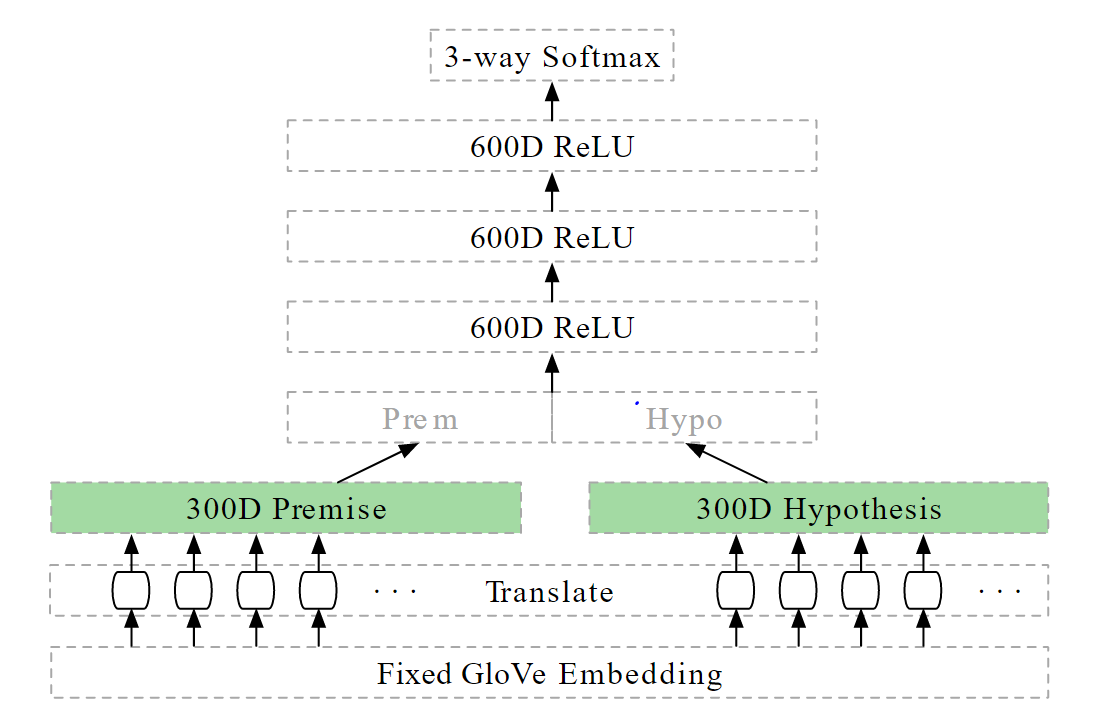
Natural Language Inference (primary task):

Model was based on this architecture:



Implemented a Keras functional model which will take 2 inputs in the form of integer sequences (1 sequence each for the hypothesis and premise sentences).

The sentences will be first vectorized using the Keras TextVectorization layer and then fed to the model. Since the model expects inputs to be integer sequences, the whole text dataset will have to be vectorized beforehand.

Then word embeddings are generated by a keras Embedding layer from these integer sequences. I have used pre-trained GloVe embeddings. After word embeddings are generated, these 2 vectors are encoded by bidirectional LSTM layers and then the resulting sentence embeddings are concatenated. These embeddings are then fed to a few fully connected Dense layers (with ReLU as activation function). Output is generated using a 3-way softmax layer.

GloVe embeddings are not updated during training. Used keras callback API to checkpoint and save the models with the best accuracy on the validation set.

Used Kaggle’s TPU to the train the model, used RMSprop as the optimizer.

Results: The model’s accuracy on the test set was 78.89%

Classifying sentence length (Auxiliary task):

Dataset downloaded from [here.](https://github.com/facebookresearch/SentEval/tree/master/data/probing)

This is a classification task where the goal is to predict the sentence length which has been binned in 6 possible categories with lengths ranging in the following intervals: --0: (5-8), 1: (9-12), 2: (13-16), 3: (17-20), 4: (21-25), 5: (26-28).

Used the previously trained model (model trained for primary task) to get word representations. Then these representations are passed to 2 fully connected Dense layers along with dropout. Output is predicted using a 6-way softmax layer.

Used a subset of the dataset to the train the probe as vectorization of the whole dataset was causing excess memory allocation.

The text dataset was vectorized beforehand for this task too.

Used Kaggle’s TPU to the train the model, used RMSprop as the optimizer.

Results: The probe’s accuracy on the test set was 0.6743

Estimating a baseline performance:

Created a classifier with the same number of dense layers as the probe and a

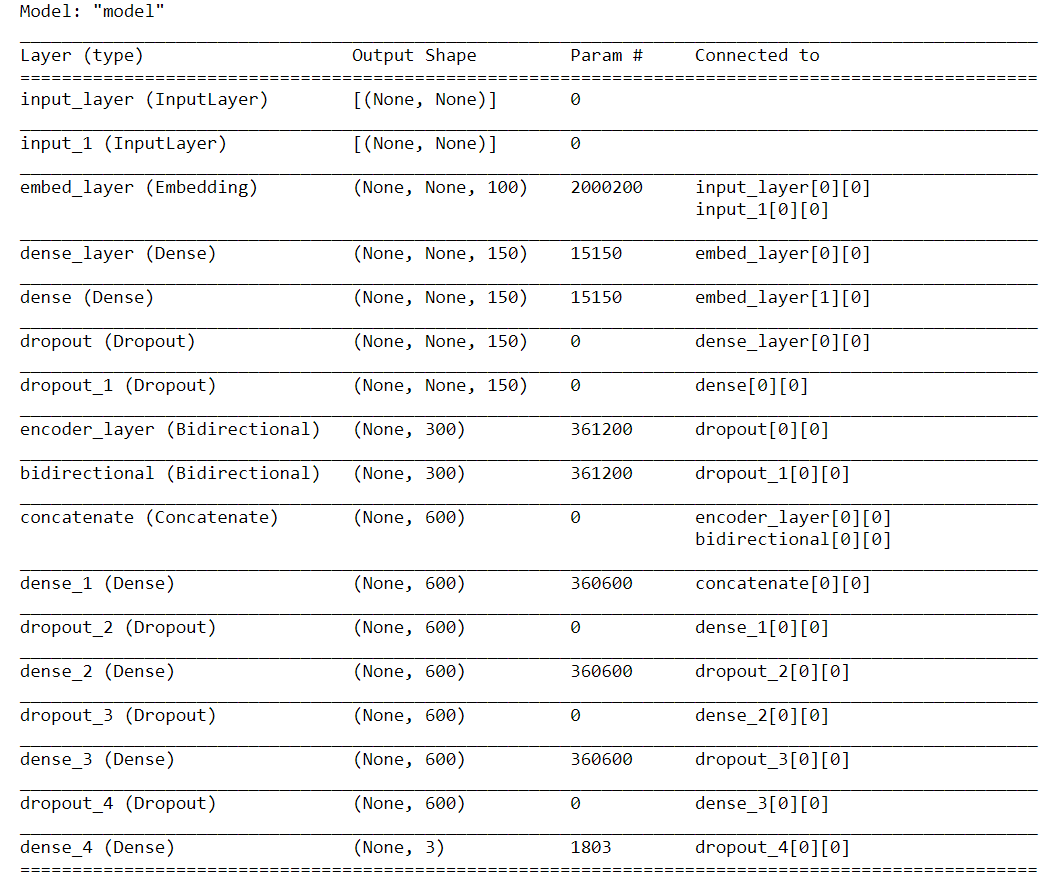
softmax layer at the end. This classifier accepts input with the shape of the word representations which were outputted by the LSTM encoder. Created a random NumPy matrix with the same shape as the word representations output by the original model.

Trained this model on the randomized word representations to estimate a baseline accuracy.

Downloaded the word representations of the test set output by the probe’s encoder layer. Used these word representations to evaluate the classifier which was trained on randomized word representations. The accuracy on the test set was 15.77%

One thing I noticed was that word representations output by the LSTM encoder were sparse. A better alternative for estimating a baseline would have been creating a random but sparse matrix to train the probe testing model.

Conclusion: Since the accuracy of the probe model (~65%) was greater than the model trained on a randomized input (~15%) we can conclude that the property of sentence length is encoded in the word representation till some extent.

Model used for primary task- 

Model used for probing: