I modify the given models in the following part:

1. In \_\_init\_\_ function of DanModel I firstly define linear2 before criterion is checked in order to pass the Gradescope test. Also in the initialization I define relu and dropout layer and build the neural network using sequence to align linear1, relu, dropout and linear2 layer in order. I also put the model to the corresponding device for later usage on GPU.
2. I define the average function by dividing the sum of embedding over the text\_len given.
3. Next function is the forward method. I input input\_text into embedding, then put embedding into average, then finally put average into the network. These steps are intuitive and in order.
4. Then I define the rule of two criterion ‘CrossEntropyLoss’ and ‘MarginRankingLoss’ in the batch\_step method in class DanGuesser. The CrossEntropyLoss is as simple as putting predictions and labels (integers after being looked up) into the nn.CrossEntropyLoss function. The calculation of MarginRankingLoss is to firstly compute the distance between predictions using samples and positive samples, then compute the distance between predictions using samples and negative samples. Input these two distance and all one vectors (this means we want positive samples to rank higher) into the nn.MarginRankingLoss function.

By the criterion I get the loss of each learning step, and I also update the optimizer here because it is an argument of this method.

1. I define the rules of counting wrong predictions in the method number\_errors. For CrossEntropyLoss I consider the column with the largest magnitude to be the predicted class, and for MarginRankingLoss I compare the string predictions with string labels.
2. I also make some small modifications for let the model suit for running on GPU based on the error reports.
3. After some trial and error, I find SGD optimizer runs so slowly and cannot match the size of the full training set. Thus, I changed to Adam optimizer which is very popular in NN today. It is faster by automatically adjusting the learning rate.

I trained the model using the following parameters and they all quickly converged to acc 1.00 within 200 epochs. I put these parameters as the default setting in parameter.py.

A screen shot of a computer code

Description automatically generated

The resulting plots are shown below:

With CrossEntropyLoss:

A graph of a diagram

Description automatically generated with medium confidence

A graph showing a line

Description automatically generated

A graph of different sizes of dots

Description automatically generated with medium confidence

With MarginRankingLoss:

A screenshot of a graph

Description automatically generated

A graph showing a line graph

Description automatically generated

A chart of different types of dots

Description automatically generated with medium confidence

The parameters that I used to train the full set are different from those in the toy training set. This is because I don’t want to make the training of toy set also to be slow as the dimensions of embedding and hidden units need to be really high.

Thus, I show the best group of parameters in the following and the corresponding accuracy and loss.

python -i dan\_guesser.py --secondary\_questions ../data/qanta.buzzdev.json.gz --questions ../data/qanta.buzztrain.json.gz --dan\_guesser\_plot\_viz viz --dan\_guesser\_hidden\_units 300 --dan\_guesser\_vocab\_size=40000 --dan\_guesser\_max\_classes=1000 --dan\_guesser\_num\_workers 10 --dan\_guesser\_num\_epochs 100 --dan\_guesser\_embed\_dim 1000 --dan\_guesser\_ans\_min\_freq=2 --dan\_guesser\_nn\_dropout 0.1 --dan\_guesser\_batch\_size=1000 --dan\_guesser\_plot\_every 1 --dan\_guesser\_criterion=MarginRankingLoss --dan\_guesser\_initialization='xaiver' --dan\_guesser\_device='cuda' --dan\_guesser\_learning\_rate=0.001

True

Setting up logging

INFO:root:Loading questions from ../data/qanta.buzztrain.json.gz

INFO:root:Read 18460 questions

INFO:root:Loading questions from ../data/qanta.buzzdev.json.gz

INFO:root:Read 1152 questions

INFO:root:Loaded 18460 train examples

INFO:root:Loaded 1152 dev examples

INFO:root:Example: {'text': 'After this character relates a story about how he didn\'t know the proper way to use a wheelbarrow, he tells of how a captain dining with his father mistakenly rubbed his hands in a punch bowl.\xa0This "sea Prince of Wales" leaves his home by hiding out in a canoe near a coral reef, and he is mistakenly called "Hedgehog" by a character who offers him a ninetieth lay, a partner of Bildad named Peleg. A door is broken down in Mrs. Hussey\'s establishment after he locks himself in his room during a "Ramadan."\xa0He is first encountered in the Spouter-Inn where the landlord thinks he may be late because "he can\'t sell his head," and his coffin helps save the narrator after the ship he\'s on sinks.\xa0For 10 points, name this native of Rokovoko and savage companion of Ishmael in Moby-Dick.', 'answer': 'Queequeg', 'page': 'Queequeg', 'category': 'Literature', 'subcategory': 'Literature American', 'tournament': 'ACF Winter', 'difficulty': 'College', 'year': 2010, 'proto\_id': '5476990eea23cca905506d51', 'qdb\_id': None, 'dataset': 'protobowl', 'qanta\_id': 0, 'tokenizations': [[0, 192], [193, 398], [399, 506], [507, 693], [694, 783]], 'first\_sentence': "After this character relates a story about how he didn't know the proper way to use a wheelbarrow, he tells of how a captain dining with his father mistakenly rubbed his hands in a punch bowl.", 'answer\_prompt': '', 'gameplay': True, 'fold': 'buzztrain'}

INFO:root:Using device 'cuda' (cuda flag=False)

INFO:root:Initializing guesser of type Dan

INFO:root:Creating embedding layer for 40000 vocab size, with 500 dimensions (hidden dimension=300)

100%|███████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 3870/3870 [00:02<00:00, 1788.99it/s]

INFO:root:Loaded 3870 questions with 500 unique answers

100%|█████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████████| 16/16 [00:00<00:00, 268435.46it/s]

INFO:root:Loaded 16 questions with 5 unique answers

INFO:root:Training KD Tree for lookup with dimension 3870 rows and 300 columns

INFO:root:[Epoch 0001] Dev Accuracy: 0.000 Loss: 1.511710

INFO:root:[Epoch 0002] Dev Accuracy: 0.188 Loss: 1.001357

INFO:root:[Epoch 0003] Dev Accuracy: 0.125 Loss: 0.785582

INFO:root:[Epoch 0004] Dev Accuracy: 0.125 Loss: 0.684031

INFO:root:[Epoch 0005] Dev Accuracy: 0.125 Loss: 0.647545

INFO:root:[Epoch 0006] Dev Accuracy: 0.125 Loss: 0.562046

INFO:root:[Epoch 0007] Dev Accuracy: 0.188 Loss: 0.475262

INFO:root:[Epoch 0008] Dev Accuracy: 0.062 Loss: 0.462414

INFO:root:[Epoch 0009] Dev Accuracy: 0.062 Loss: 0.419212

INFO:root:[Epoch 0010] Dev Accuracy: 0.125 Loss: 0.400569

INFO:root:[Epoch 0011] Dev Accuracy: 0.062 Loss: 0.364754

INFO:root:[Epoch 0012] Dev Accuracy: 0.125 Loss: 0.335960

INFO:root:[Epoch 0013] Dev Accuracy: 0.062 Loss: 0.312166

INFO:root:[Epoch 0014] Dev Accuracy: 0.062 Loss: 0.300576

INFO:root:[Epoch 0015] Dev Accuracy: 0.062 Loss: 0.302814

INFO:root:[Epoch 0016] Dev Accuracy: 0.188 Loss: 0.278684

INFO:root:[Epoch 0017] Dev Accuracy: 0.000 Loss: 0.270702

INFO:root:[Epoch 0018] Dev Accuracy: 0.188 Loss: 0.262146

INFO:root:[Epoch 0019] Dev Accuracy: 0.125 Loss: 0.224272

INFO:root:[Epoch 0020] Dev Accuracy: 0.000 Loss: 0.237669

INFO:root:[Epoch 0021] Dev Accuracy: 0.000 Loss: 0.235516

torch.OutOfMemoryError: CUDA out of memory. Tried to allocate 6.56 GiB. GPU 0 has a total capacity of 8.00 GiB of which 0 bytes is free. Including non-PyTorch memory, this process has 17179869184.00 GiB memory in use. Of the allocated memory 8.89 GiB is allocated by PyTorch, and 7.48 GiB is reserved by PyTorch but unallocated. If reserved but unallocated memory is large try setting PYTORCH\_CUDA\_ALLOC\_CONF=expandable\_segments:True to avoid fragmentation. See documentation for Memory Management (https://pytorch.org/docs/stable/notes/cuda.html#environment-variables)

I stopped here because my PC doesn’t allow me to run this huge set further. The loss keeps decreasing so I believe if memory is allowed the accuracy can be further improved from the current highest 0.188.