Previous:

* Followed 2 tutorials on creating tensorflow chatbot
* 1) Simple DNN network on small dataset picking random answer from tag as response
* 2) Large Reddit dataset that uses nmt (neural machine translation) model trained on 100000 pairs + to create responses.
* Downloaded amazon QA dataset –

6/03

* Follow tensorflow tutorial on text classification

From the tutorial: Build a tf.keras.Sequential model and start with an embedding layer. An embedding layer stores one vector per word. When called, it converts the sequences of word indices to sequences of vectors. These vectors are trainable. After training (on enough data), words with similar meanings often have similar vectors.

This index-lookup is much more efficient than the equivalent operation of passing a one-hot encoded vector through a tf.keras.layers.Dense layer.

A recurrent neural network (RNN) processes sequence input by iterating through the elements. RNNs pass the outputs from one timestep to their input—and then to the next.

The tf.keras.layers.Bidirectional wrapper can also be used with an RNN layer. This propagates the input forward and backwards through the RNN layer and then concatenates the output. This helps the RNN to learn long range dependencies.

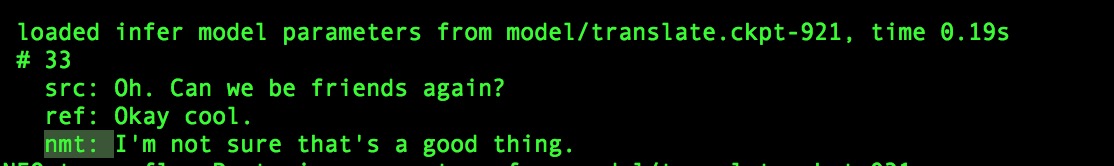
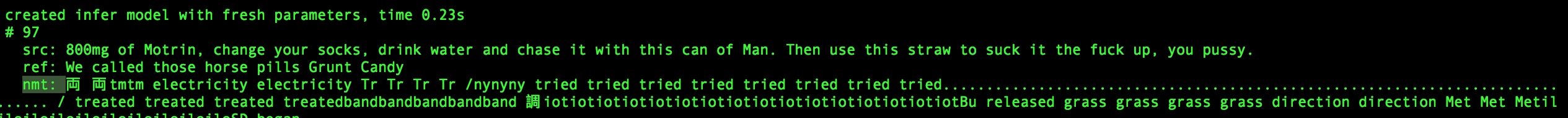
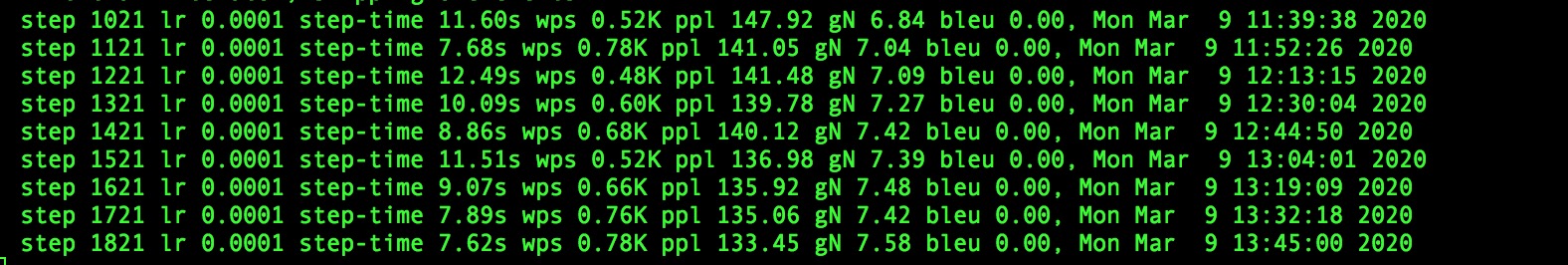
7/03

* Look into reddit data to create conversational dataset with multiple threads from a conversation
* Installed bigquery command line to read the data
* Running dataflow command cause d recursive error…
* Fixed by running outside virtualenv, installing pip
* curl [https://bootstrap.pypa.io/get-pip.py -o get-pip.py](https://bootstrap.pypa.io/get-pip.py%20-o%20get-pip.py)
* sudo python get-pip.py
* from here ran pip install apache-beam[gcp]
* ran the dataflow script outside virtualenv – now up and running, constantly checking quota limits to keep an eye on costs not
* wanted to extract score for each comment in reddit data. added score to createdata.py
* changed data structure of JSON to be nested and clear with scores
* made reddit dataset really small on big query for testing for structural purposes

8/03

* follow reddit tutorial
* cloned github repo locally
* tutorial did note that running locally would take ‘forever’ to train
* used their data (as it was more than I had…. And claims to be ‘small’ amount of data)
* upgraded tensorflow, uninstalled tensorflow, installed tensorflow 1.14 – worked

9/03

* running locally during first epoch of training, every 100 steps took on average 20 minutes. Epoch finishes after 1000 steps. Nmt produces example outcomes, varies of ok response to not very good response to context
* 
* can see from this nmt bot response varies
* epoch 2 steps through quicker:
* 

learnt about subwords - <https://medium.com/@makcedward/how-subword-helps-on-your-nlp-model-83dd1b836f46>

learnt about learning rates <https://machinelearningmastery.com/understand-the-dynamics-of-learning-rate-on-deep-learning-neural-networks/>

Learnt about RNN and LSTM <https://www.youtube.com/watch?v=UNmqTiOnRfg> , <https://www.youtube.com/watch?v=UNmqTiOnRfg> (see Intro to RNN doc for notes)

# Received confirmation of free TPU

Followed this tutorial:

To start a cpu instance;

ctpu up --zone=us-central1-b  --tf-version=2.1

to kill it:

ctpu delete --zone=us-central1-b

To start gcloud shell: gcloud alpha cloud-shell ssh

To then connect to the VPU: gcloud compute ssh mattprankin --zone=us-central1-f

Can’t use the tutorials code with TPU – it isn’t written for TPU. We couldn’t even run dan’s code on the Cloud Compute VM that we span up because those machines come with python 3.5 and we need 3.6 for his tutorial.

Good news – our model finished, so we can continue following his tutorial for now. Sadly, the output wasn’t very good – all responses looked like:

I'm not sure that I'm not sure.  
I'm not sure that I'm not sure. I'm not sure if I'm not sure about that.  
I'm not sure that's the only one I'm not sure.  
I'm not sure if I'm not sure about that. I'm not sure if I'm not sure about that.  
I'm not sure if you're right.  
I'm not sure that's a good idea.  
I'm not sure if you're right.  
I'm not sure that was the best thing I'm not sure.  
I'm not sure that's the same thing.  
I'm not sure if you're right, but I'm not sure if I'm not sure about that.  
I'm not sure if I'm not sure about that.

Next step – try running on a powerful instance in the cloud where we can increase the vocab size

Learning from: <https://www.youtube.com/watch?v=gFxiQXnt9w4&list=PLQVvvaa0QuDdc2k5dwtDTyT9aCja0on8j&index=8>

He talks about the different metrics to tweak to improve bot

Tensorboard

Bleu score best determining factor for translations

For chatbot for a given comment there’s A LOT of responses.. so bleu score isn’t going to be too high

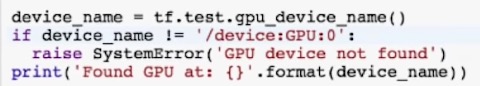
Training on 5000 epochs likely to overfit!

NEXT STEPS:

1. Spin up powerful Compute Engine instance, connect with ssh and try running the same thing that you ran locally and see how long it takes. Be wary of the costs!
2. Explore using GPT2 : <https://minimaxir.com/2019/09/howto-gpt2/>
3. We have free TPU resource. Follow some tutorials from <https://google.github.io/seq2seq/nmt/> to understand how seq2seq works
4. Try to build own model –
5. Try use tpu – call tensrflow functions with a flag

11/03

learned about GPU’s TPU’s <https://www.youtube.com/watch?v=tCYSce6l8gA>

* to train on mass amounts of data efficiently – need to use specialised software – gpu(graphic processing units)/tpu(tensor processing units)
* 

21/03

ran through this <https://blog.keras.io/a-ten-minute-introduction-to-sequence-to-sequence-learning-in-keras.html>

* ran into errors

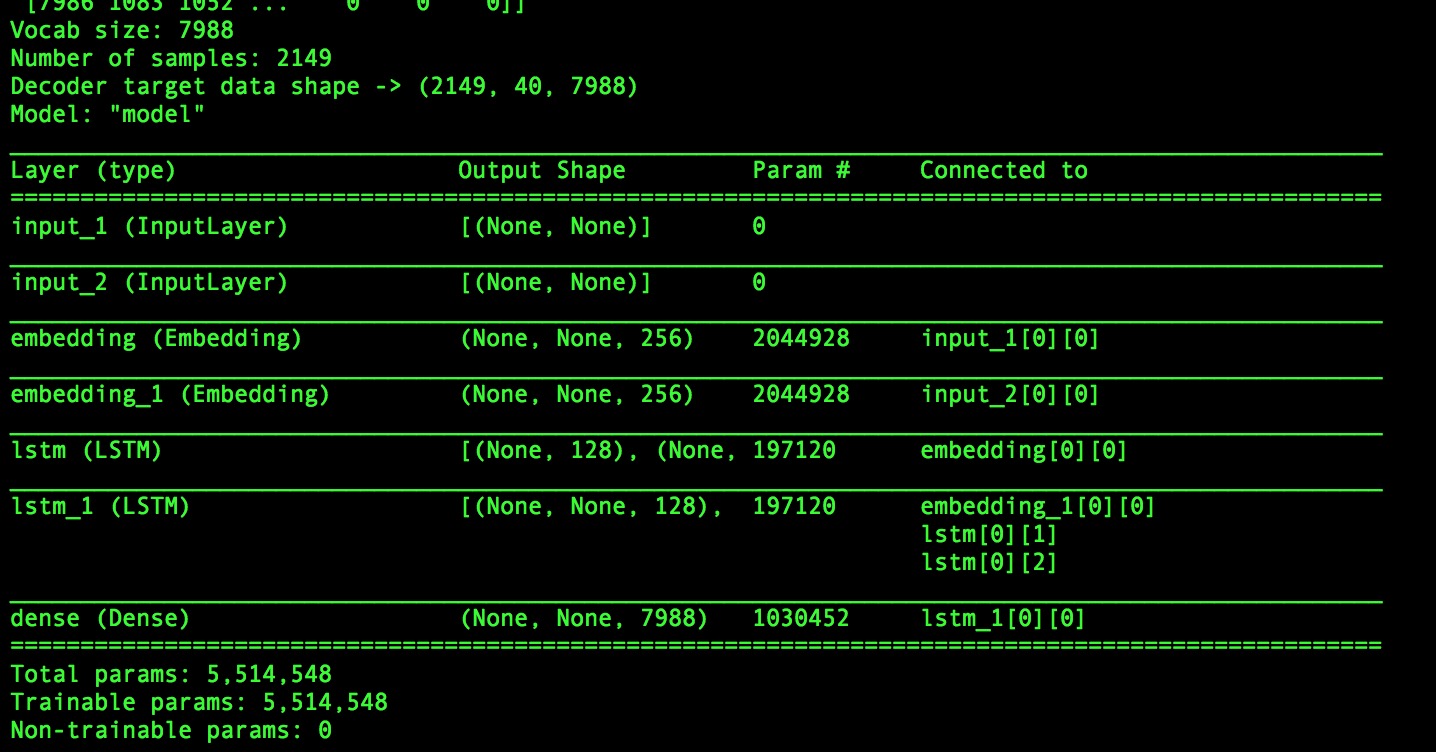
found new tutorial <https://colab.research.google.com/github/tensorflow/examples/blob/master/community/en/transformer_chatbot.ipynb#scrollTo=TpR7kz4jFkPJ> - transformer

* ran locally and took forever

new plan to move to colab following data input from here: <https://colab.research.google.com/notebooks/io.ipynb#scrollTo=P6aWjfTv7bit>

* use GCS on colab

22/03

* followed the about tutorials and merged them and locally ran on test data (2100 lines q’s and a’s)
* results not great
* 
* thinking about uses masses of data set up gpu instance on cloud
* set up VM following tutorial <https://blog.kovalevskyi.com/migrating-our-chatbot-training-logic-from-colab-to-google-cloud-engine-d34b788e8eff>
* requested in cloud increase gpu all regions limit to 1 in order to create instance
* tested new instance on nmt code that I ran locally a few weeks back and it ran in <10mins

<https://medium.com/predict/creating-a-chatbot-from-scratch-using-keras-and-tensorflow-59e8fc76be79>

= <https://colab.research.google.com/drive/11os3isH4I4X76dwOAQJ5cSRnfhmUziHm#scrollTo=2Mfco9WKukhS&forceEdit=true&sandboxMode=true>

* this model used for latest model attempt

Next steps:

1. creating python script to read files in cloud bucket and write new files for all data extracted into bucket a few weeks ago
2. create mass dataset to use for chatbot
3. spin up my VM instance and run my code with massive dataset and see if performances improves
4. Improve performance by changing vocab size and other params

Following <https://cloud.google.com/appengine/docs/standard/python/googlecloudstorageclient/read-write-to-cloud-storage>

* Reading and writing into GCS

24/03

* Created script that will read all files in GCS bucket and write out massive train and test files to the VM

Tomorrow morning jobs:

* Move files needed to run my script into github venvfyp directory - done
* Push to git - done
* Go on VM clone repo
* Run prepare data
* Train

25/03

* Moved code to venvfyp folder
* Pushed to git
* Call with supervisor
* Tried running code on train data locally
* Error occurred !!
* Tried fixing

To do:

* Fix error so train.py runs
* In VM git clone repo
* Run prepare data wait for new train.py files to be made
* Run train.py

Went to spin up instance received error of The zone 'projects/chatbotfyp-1/zones/us-central1-a' does not have enough resources available to fulfill the request. Try a different zone, or try again later.

27/03

* Changed prepare data script to use up less RAM usage but rather than loading each files contents into a stored array which is massive. It iterates through each file downloaded and dumps data into new files then closes the file
* Ran on VM and test/train from and to files got created
* Made a bash script for uploading files from machine to the GCS. Ran this is VM to store new converational data files
* Tried running train file on VM – first run got ‘killed’ second hung - presume RAM memory issue
* Max samples = 2000 for testing

# Received tensorflow error [Tensorflow Allocation Memory: Allocation of 38535168 exceeds 10% of system memory](https://stackoverflow.com/questions/50304156/tensorflow-allocation-memory-allocation-of-38535168-exceeds-10-of-system-memor)

* Fixed by changing batch size from 256 to 3.

Test data for out of sample validation –

Jobs for tomorrow:

Work out how to get git on VM

28/03

* Spun up new powerful instance
* Had to install git and python3.6 as using keras and tensorflow
* Install requirements.txt file
* Found a way of creating instances that had packages for tf usage pre installed. Saved time

Gpu instance wasn’t fast: nvidia v100 – 1gpu – 3.75gb mem – 1cpu

Attempt new instance – nvidia v100 - 1 gpu - 15gb mem – 4cpu –

* Using gpu as device
* Per epoch 145s 74ms/sample
* Batch size 3
* Using cpu as device
* Per epoch 122

Tried an even bigger machine with lots more memory.

* Sample size 20000, max length 20 – sample went down to 7000
* Loss down then back up – learning rate too high

Keras model had to take 3d array on the to.categorical method, being the number of samples, vocab size and max length of sentence. This came to 200000\*8000\*40 which when even on super computer was far too much memory being stored. – scraped this model as it was meant for much smaller data types

Rewatching sentdex <https://www.youtube.com/watch?v=gFxiQXnt9w4&list=PLQVvvaa0QuDdc2k5dwtDTyT9aCja0on8j&index=8>

* Padding = making the input always equal the same as the longest input with padding as 0 nodes.
* This doesn’t train and perform well though as the longer the sentence the later words don’t mean anything as model learns from the shorter sentences and once in a while the longer sentences come in and don’t perform well

29/03

* Ran nmt code locally and gcloud gpu instance 1gpu 30gb memory using data from the repo. data 117k input train files
* Took 5/6 mins per 100steps with vocab size 15,000
* Ran on 100,000 took >14mins per 100steps
* Running same thing locally takes ~14mins per 100 steps

Now increasing GPU quota to run on multiple GPUs as looking to find ways of increasing training time. Applying for 8 GPUs

Reran on initial instance created where 100 steps ran every 30s – runs this way again

Now on this instance run with my data – download gcs files

Note times

Increase vocab size

Then try running on multiple GPUs – more expensive so be wary of costs

Next steps:

Run it on more memory machine if still not work try and find out

Now while epoch is running on 30gb machine….

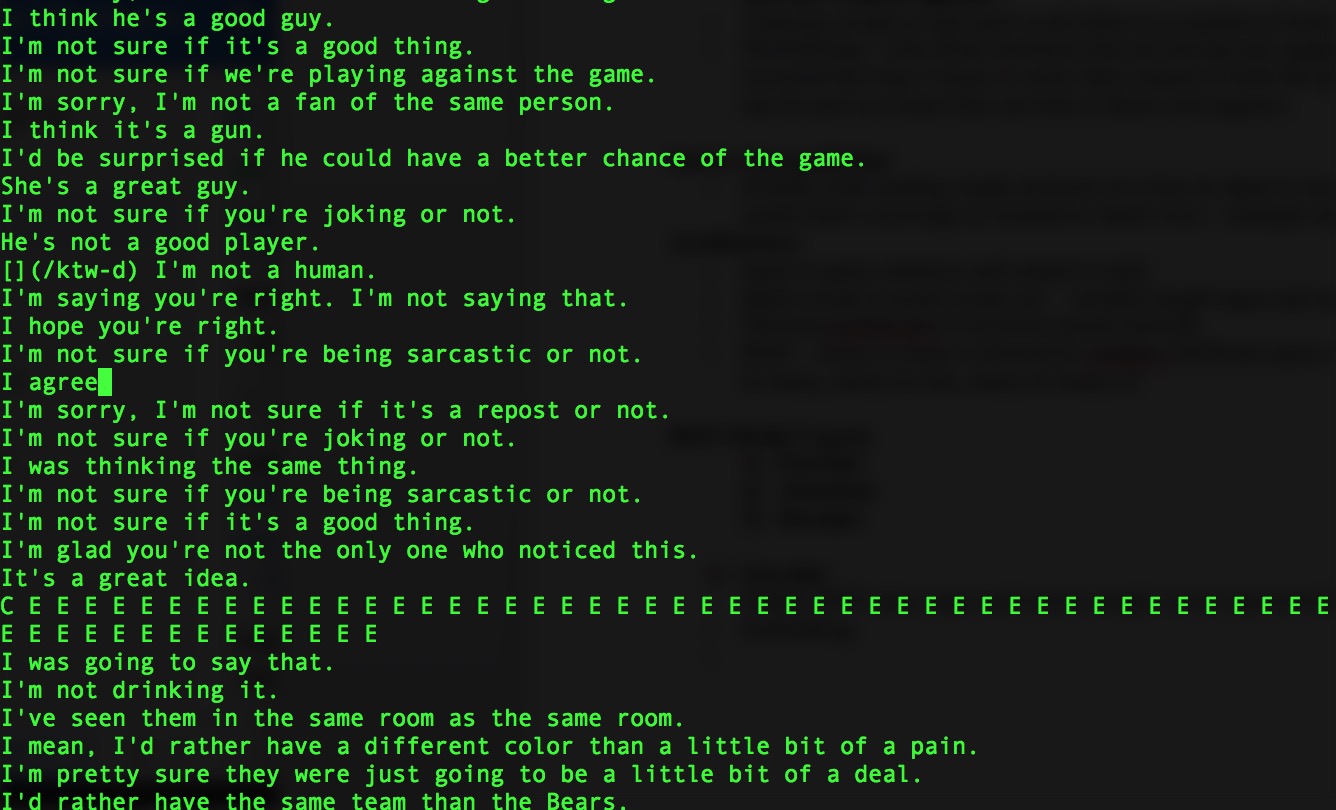
Spin up new machine which ran small data steps in 30s and test to see what params change what for the speed of training

Running on step 14000 on large dataset in epoch 1: proof that running on gpu

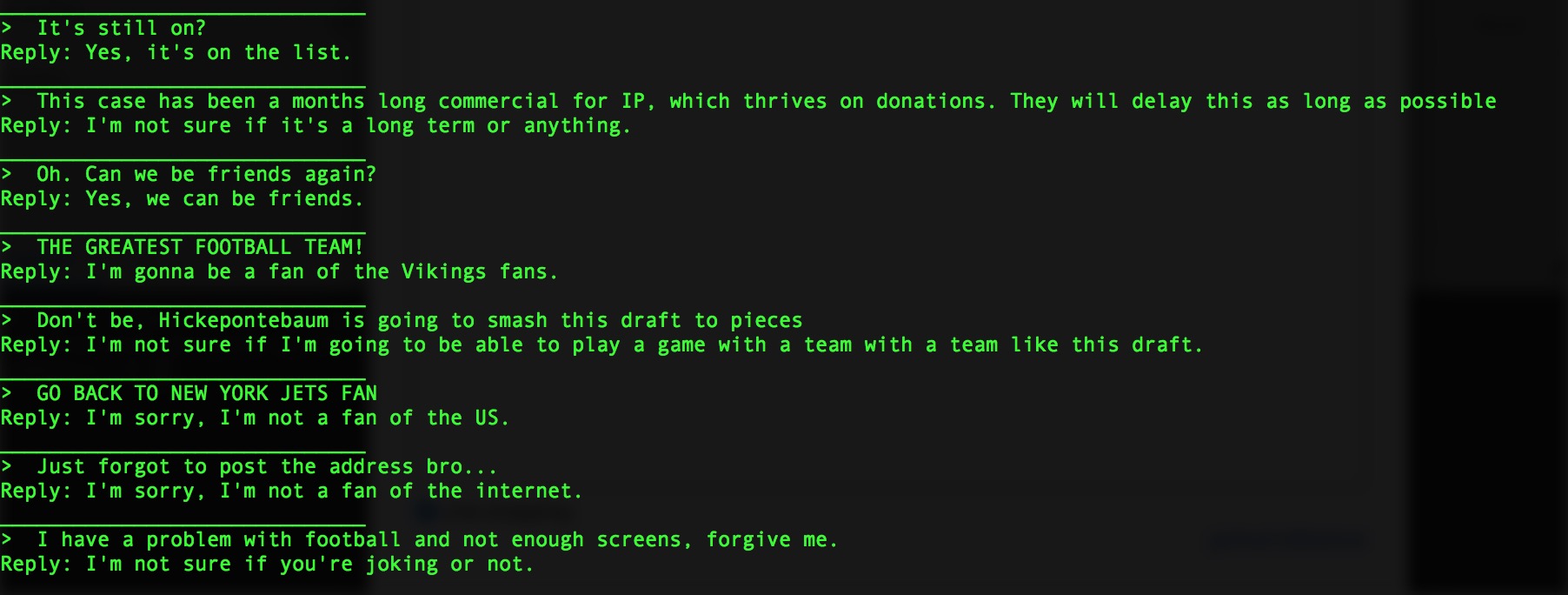
2020-03-29 22:02:04.119372: I tensorflow/core/common\_runtime/gpu/gpu\_device.cc:1325] Created TensorFlow device (/job:localhost/replica:0/task:0/device:GPU:0 with 203 MB memory) -> physical GPU (device: 0, name: Tesla V100-SXM2-16GB, pci bus id: 0000:00:04.0, compute capability: 7.0)

outputs are readable!

Examples of outputs:



outputs with the testing inputs….

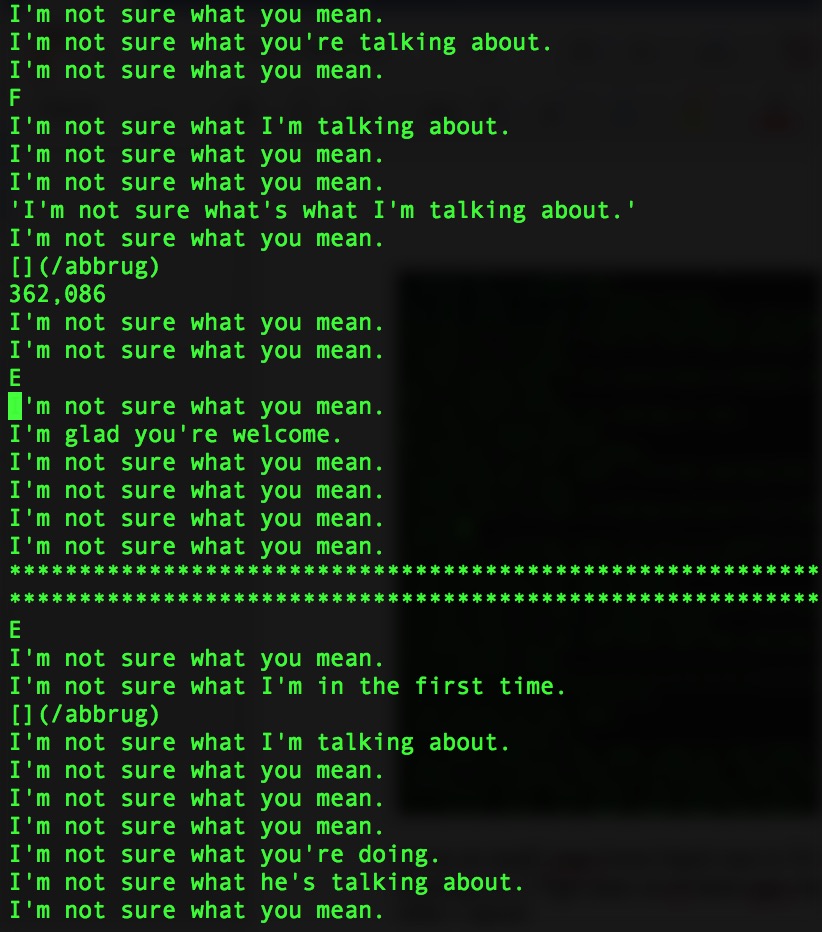


answers are pretty good! This is after 22,000 steps

to run on small vram lower batch size to 32 maybe lower

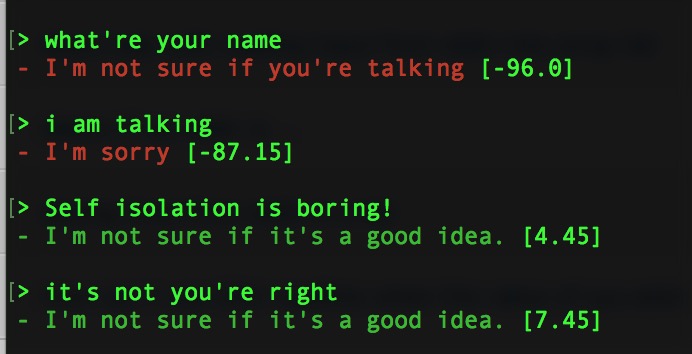
Running on 3.75gb mem on tf-latest-gpu machine batch size 32:

After 1 epoch



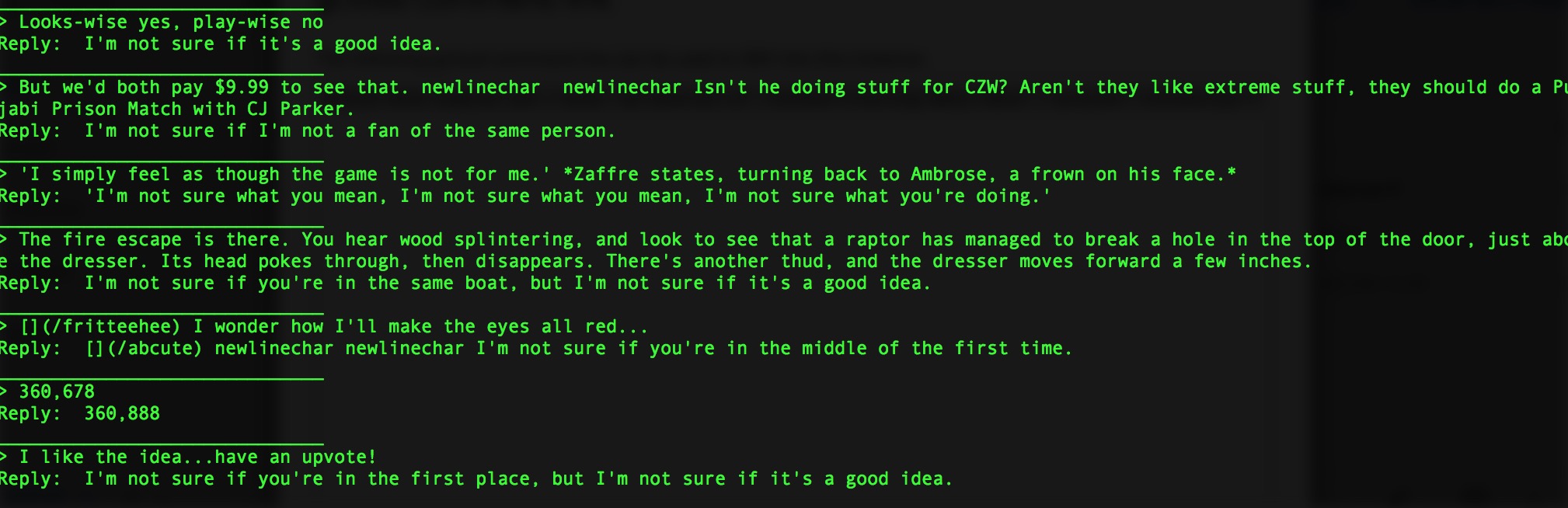
Answers not very inventive, bot seems to not know much – vocab size issue probably?

Batch size 16 – 100 steps 22/23s 3.75gb mem – ran 3 full epochs though



kind of works but still not very good..

Wrote script that pairs the output dev of the current epoch with the testing file for visualisation of performance – uploaded to GCS to be able to copy down to VM.



response not really that good and repetitive.

Try higher vocab size: tried upping vocab size 30k on 3.7fgb mem… didn’t run epoch 2 due to memory error:

Results after epoch 1 not great….



while instance region is down, added to pairing inputs and dev output script to pair the latest 5000 model of outputs with the test set. Rounding the current number of samples

**Overview of day 30/03:**

Tested different machines with different batch sizes and vocab sizes

Think that memory errors are because cuda stores data even when not using that model anymore or something

Deleted powerful instance and spinning up new powerful instance to run the last ‘out of memory’ issue

To do:

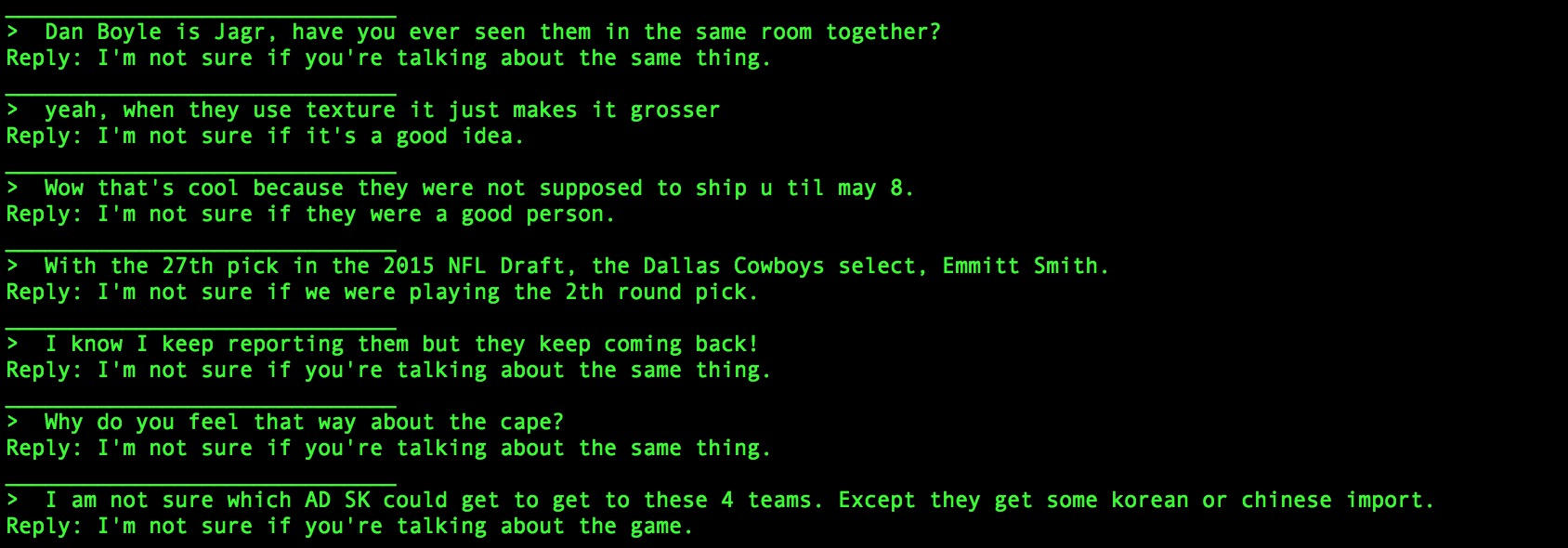
Train on powerful machine – SMALL dataset to see actual outcomes of changing params as memory issues on small machine

* Doing this to find quickest way of training efficiently

How to save our models:

Change checkpoint file to the model you want saving, number corresponds to the three files of the model you want eg for these: translate.ckpt-2763.data-00000-of-00001/translate.ckpt-2763.index/translate.ckpt-2763.meta. Checkpoint file change to 2763.. + you want hparams file

50K VOCAB 11K CORPUS 64 BATCH SIZE



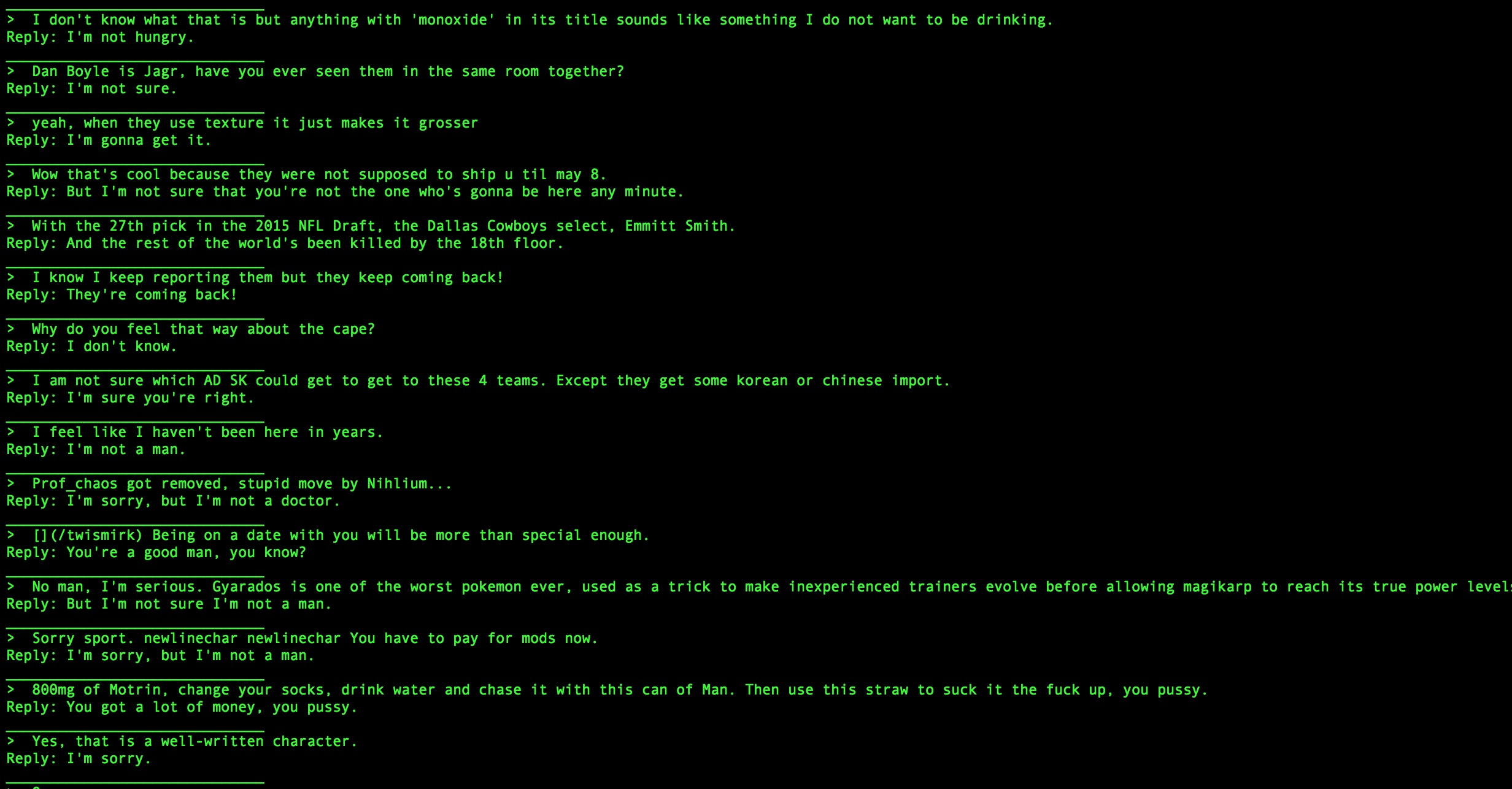
REPETITIVE

**Try running nmt code on new dataset**

* Subtitles code – dataflow command upload to GCS create JSON files
* Recursion error python version, python 3.7 can’t deal with infinite recursion
* Dataflow script created same test and train files in gcs, used same data manipulation script to create out train.from/to and test.from/to files for the nmt bot

New dataset: 6 million pairs.

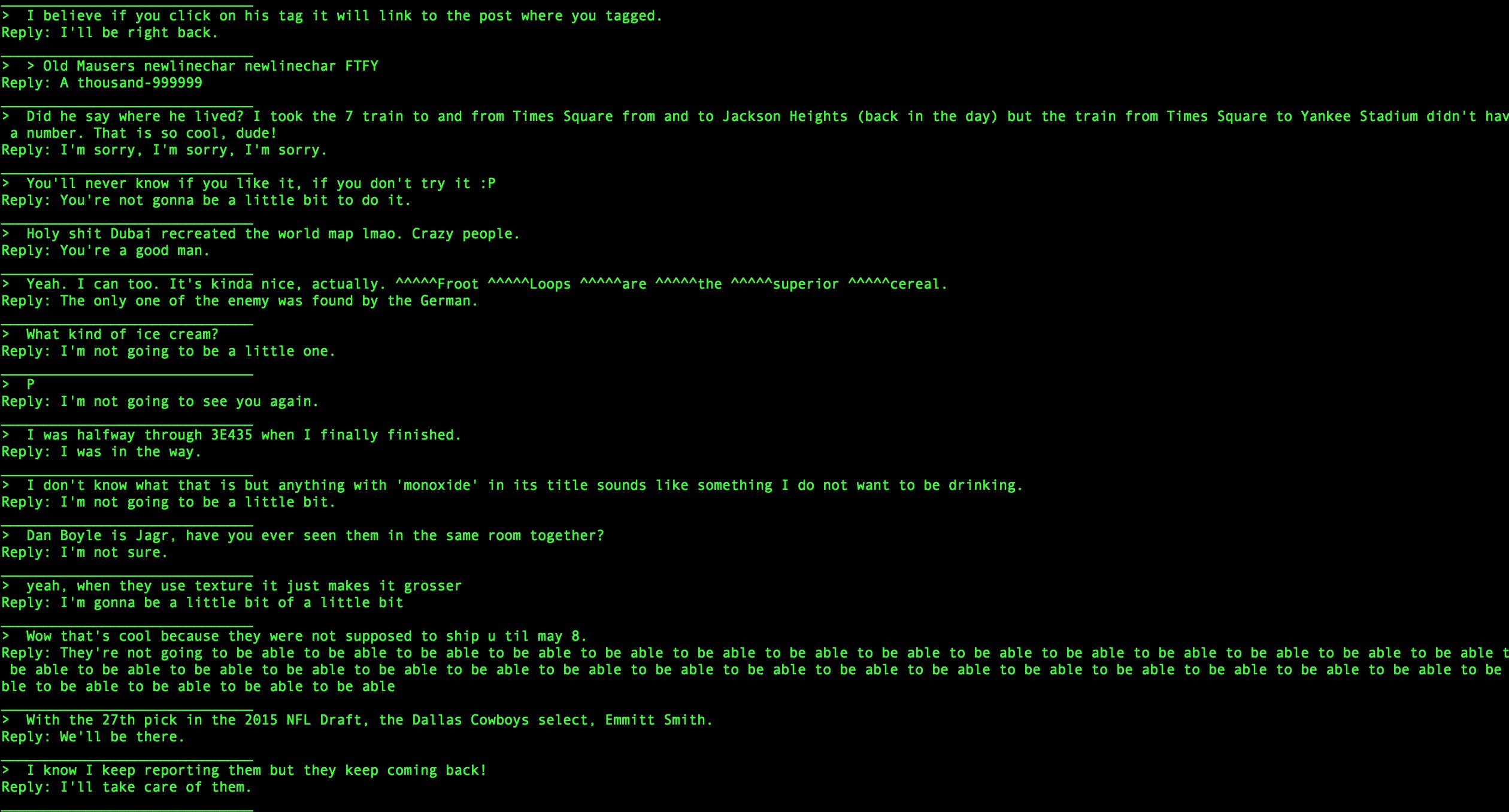
First test: 6mil pairs 128batch size 100,000 vocab ran till 20000 steps first epoch:



answers kindof make sense, has a bit of character to him ‘you pussy’

quite a lot of ‘im not a man’

Vocab 52k – Open subtitles 200k corpus – 128batch size



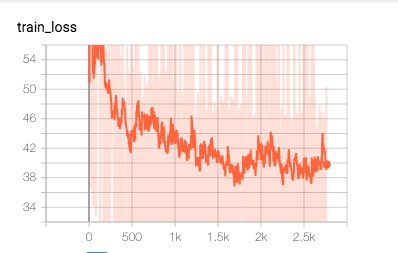
02/04:

Spinning up cheap GPU instance, run 117k open subtitles and reddit and co mpare with tensorboard.

Open subtitles 117,859 pairs 15000 vocab size x Reddit 117,859pairs 15kvocab:

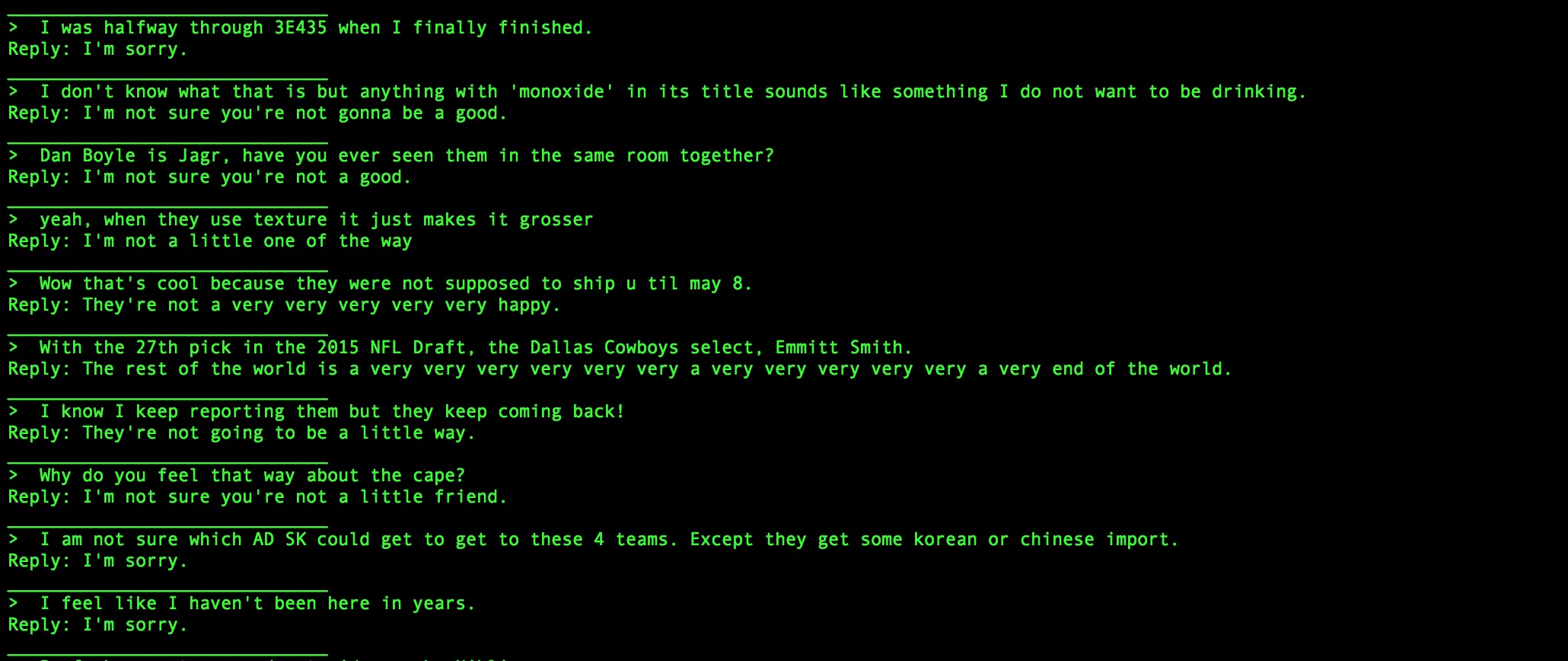


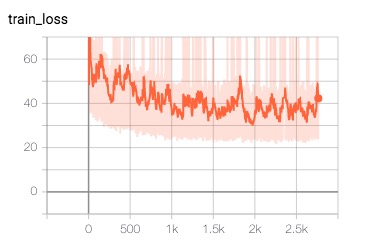
reddit^



open subtitles^

tried gnmt model hparams – plus 2 gpus… was very slow! 96s per 100 epochs… grrrrr

Changed params to 4 layer model with 4 layers enc and dec: ran way quicker but results not that good 



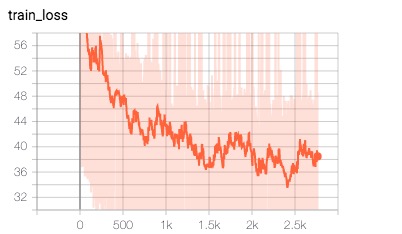
reached a low of 31.. slightly lower than that of the 2 layer model

Started fiddling in hparams for the infer mode – sample/beam/greedy

117k 36k vocab, infer-mode=sample… answers much more human like/ getting there. Temperature = 1

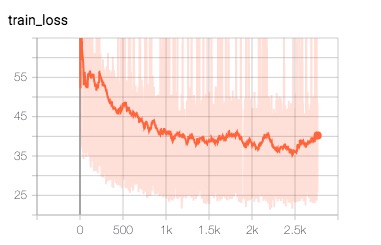


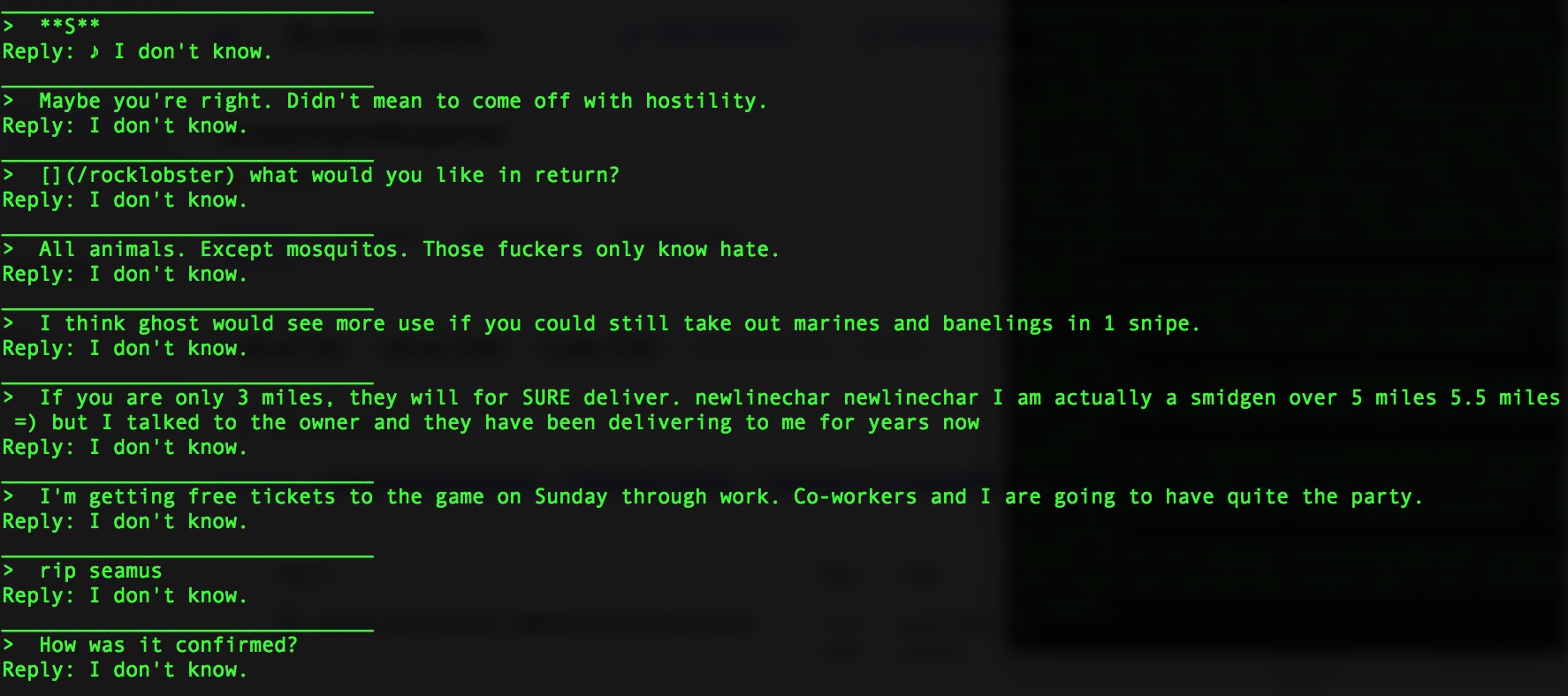
answers a bit all over the place though!



From Google meana paper: “We observe that large values of T favor contextually rare tokens, such as relevant entity names, but 13https://github.com/tensorflow/ tensor2tensor might also assign too much probability to incorrect tokens depending on the model’s predictions. Meanwhile, smaller values of T favor more common words such as articles or prepositions, which are safer but less specific.”

* Test temp 0.5,1,2
* Temp 1 = complex responses but not relating to question – possibly needs bigger corpus size and longer training.

Temp 0.5: answers all over the place.. 

Beam search width 30, 117k corpus vocab 39k… very repetitive responses of ‘I don’t know’ 

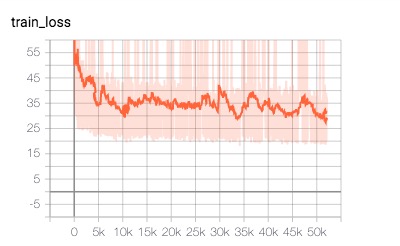
SO NOW:

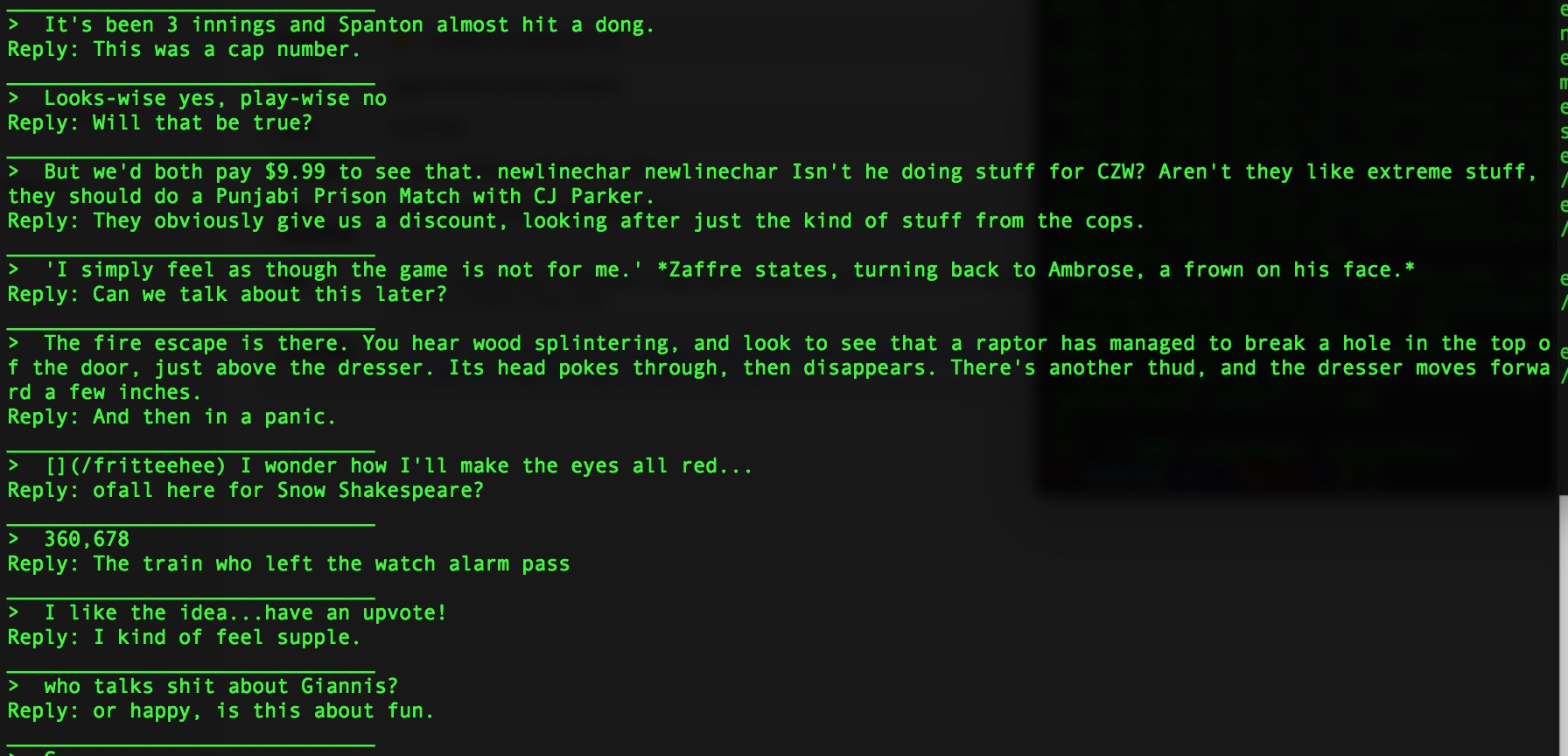
Testing on big dataset 6mil open sub – sample temp 1 – run for 22000 and compare with previous large dataset runs to 22000

NEXT:

Look to make sanmple\_softmax >0 to speed things up?

On large dataset 6mil sample infer mode temp 1 – ran for 52k steps:



training loss plateaued after about 10k steps….. probably not worth running model to 50k… 

answers were;n’t really related to the question and therefore didn’t make any sense.

Now play around with dropout values: start with dropout 0.4 ..

DATASET CHANGE:

Looking to include all contexts from a dialog – eg context/x

Only taking a context if it has a “?” in it..

Only taking questions that are shorter than 40 characters long

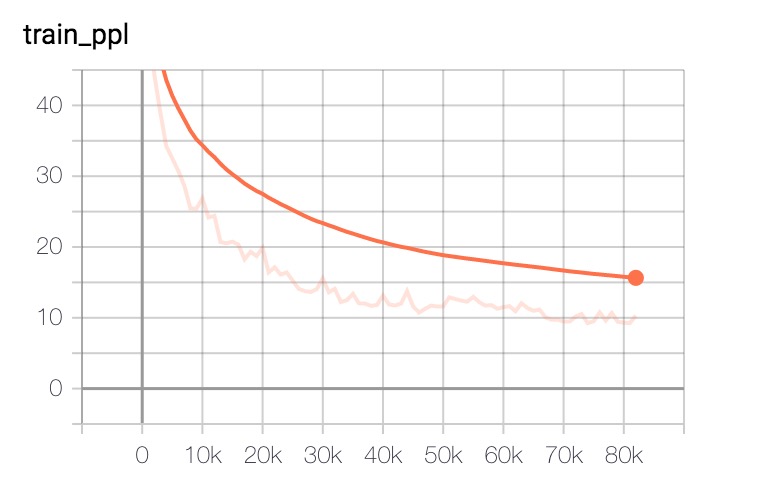
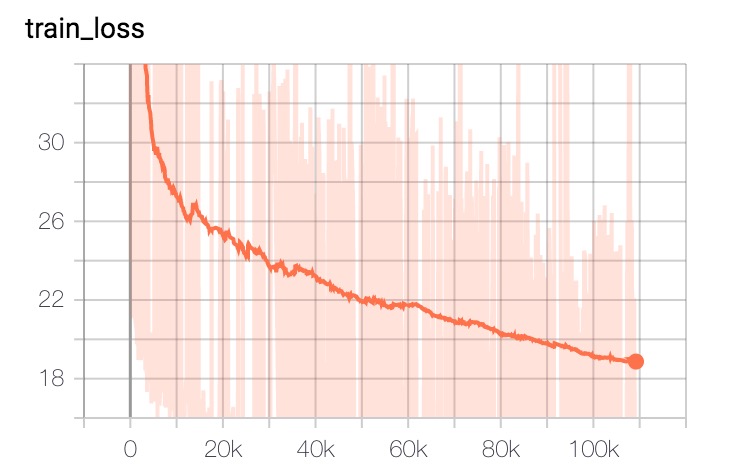
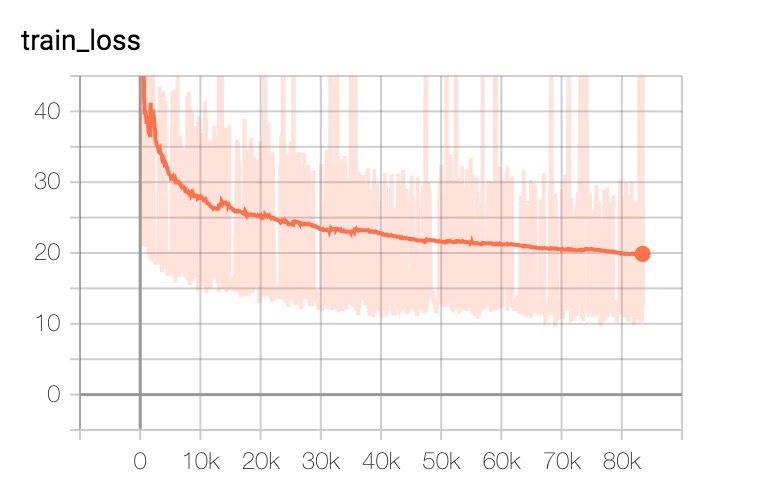
Make a more question and reponse type dataset

Ran on machine 15GB 1GPU was really slow ~50s per 100steps

Now running on 4GPUS 30GB just to see if 100 steps is quicker - not quicker!

Ran on new dataset with only questions restricting questions size to 40 chars. Running this all the way through as model is looking to be good! Same params as when running 22k steps on dataset without only questons and only context and response… 6 mil pairs. This dataset performs way better – **greedy** search, 2 layers, 0 softmax, **attention** = normed\_bahdanau, **architecture** = standard which is attention model

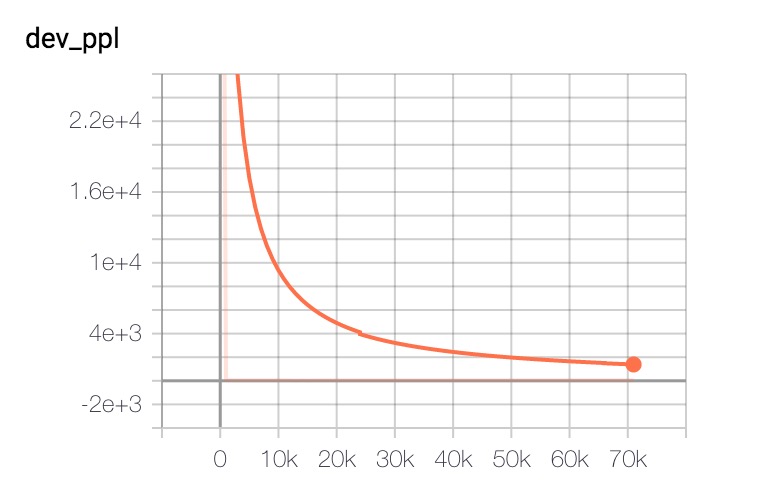
Epoch1:

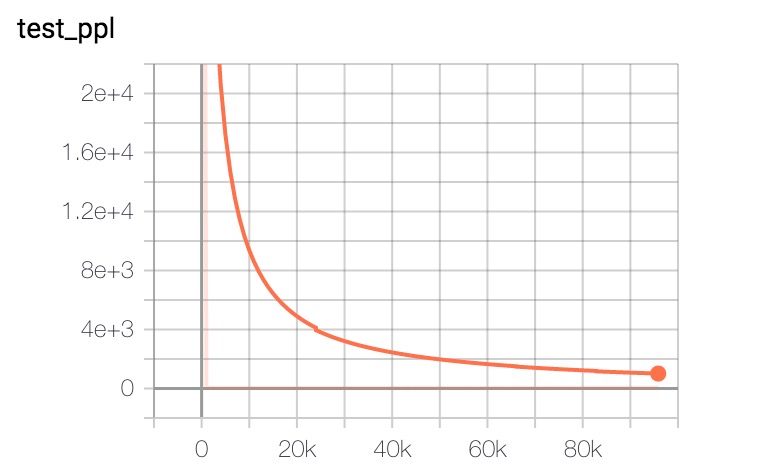


Both train loss and train ppl still falling as we hit epoch 1 at 82852 steps.

Epoch 2:

3 models in the code: gnmt/attention\_model/nmt\_model. As model above is training, want to play around with attentions and





**Adding personality:**

**Created profile detector:**

Binary text classifier – initially tried using SPAM or HAM classifier using bayes Theporem of probability P(A|B) = P(B|A)P(A)/P(B) - <https://towardsdatascience.com/spam-classifier-in-python-from-scratch-27a98ddd8e73>

Results produced were poor: for any input single letter or a false work eg: ‘g’ ‘dsfghf’ the classifier would predict as True.. Due to it being a Spam classifier and this is likely to be Spam receiving an email like ‘g’ or ‘dfghj’.

Moved on to creating a deep learning classifier using keras. Tested a LSTM model, results poor as LSTM are for sequence learning.

Model consists of embedding, 2 fully connected Dense layers with relu and sigmoid activations. Binary crossentropy loss and adam optimizer.

Results are good!

What is your name? : 0.916565

Optimising NMT:

Used original 3 epoch run as base run:

Batch size: 128

Dropout: 0.2

Encoder layers: 2

Decoder layers: 2

Infer mode: Greedy

Num units: 512

Attention: Normed bahdau

Attention architecture: Standard nmt

5 tests:

1 Dropout 0.7

2 Dropout 0.5

3 Scaled-LUONG attention – attitude.

4 Luong attention

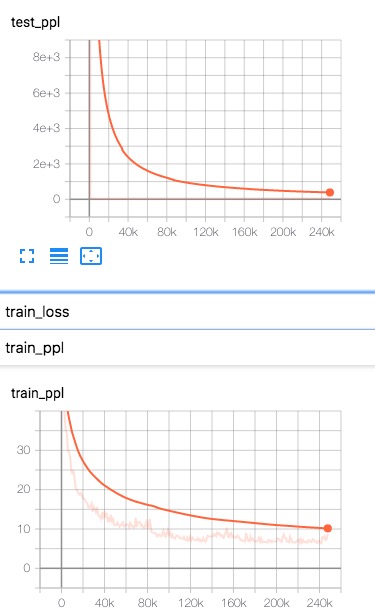
5 Bahdau attention

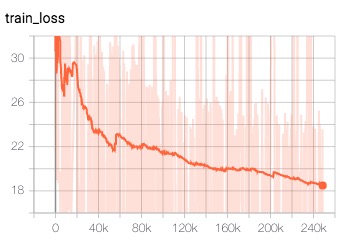
Extracted responses from all test questions and models. Created scoring system to mark which is overall best performing models to carry out next set of tests.

After scoring on Naturalness, Sensibleness, Specificness each category 1-4, f

* Naturalness this was simplty looking at the reponse alone seeing if it was natural for the English language. Down marks for short answers and poor grammatical errors and readability
* Sensibleness, scored from illogical response to very valid response
* Specificness, scored from not cotext specific 1 to context specific 4. This metric made it so very high scoring sensible models may be poorly marked on specificness due to ‘I don’t know’ or ‘im not sure’ responses

Scaled luong tensorboard:





Luong: