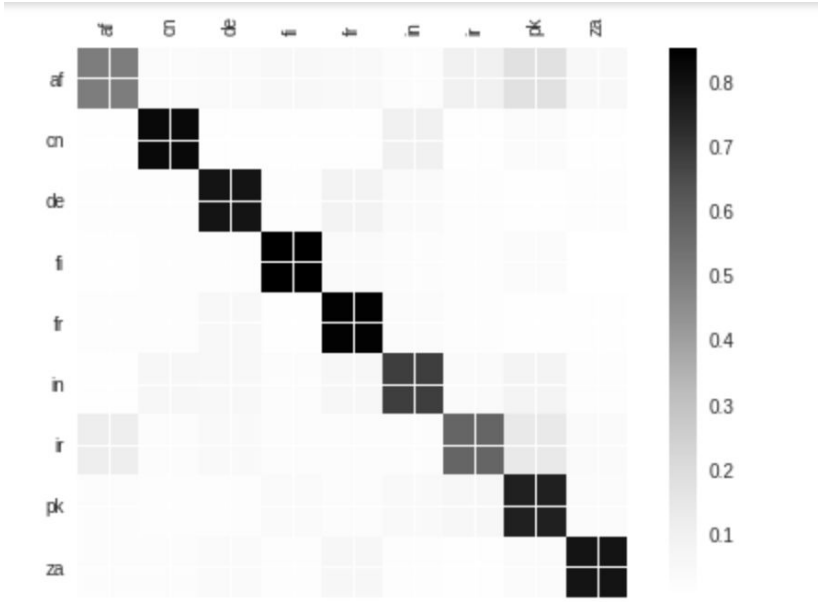
PART 2:

Description of our model:

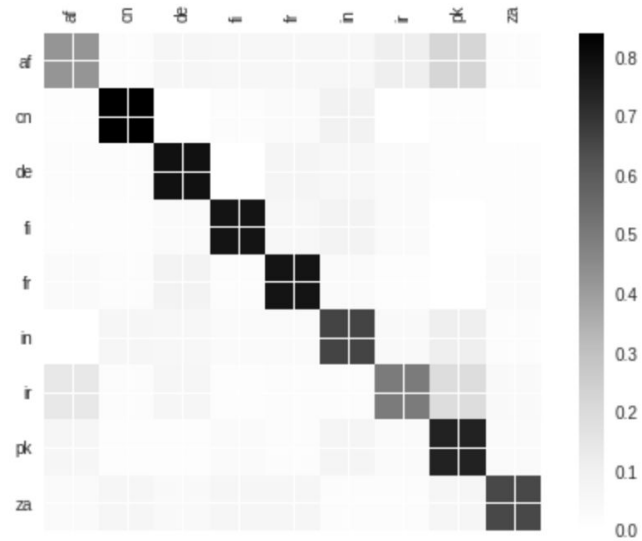
Our model is a LSTM with one input and one hidden layer of size 128. We apply softmax. We use NLLLoss for our loss and SGD with learning rate 0.004 for our optimization. We ran it for 450K epochs:

Training:




```
Training acc is  0.7362592592592593
```

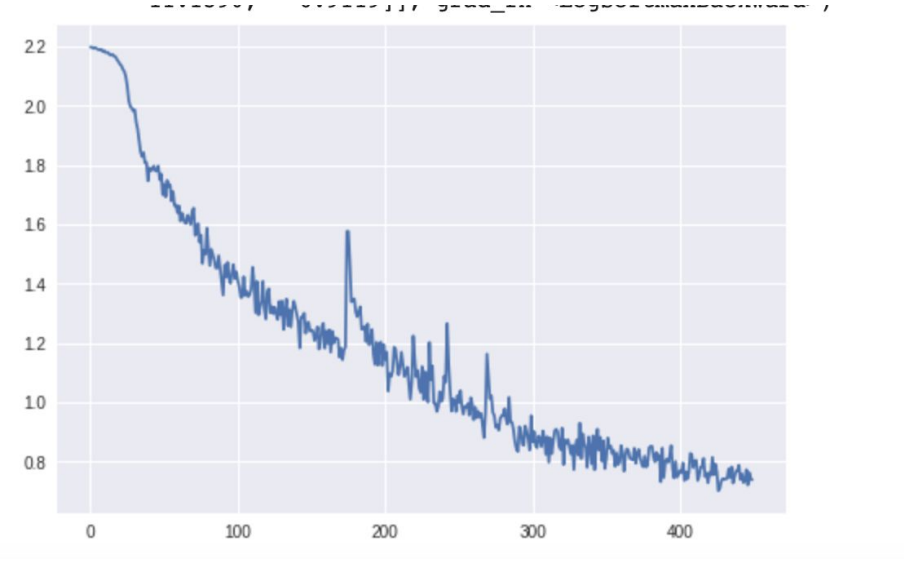
Validation:




```
Validation acc is  0.6844444444444444
```

We realized that our model is making the most mistakes on while predicting the labels associated with ‘af’

Loss:



Learning Rate Experimentation:

We started our experiment with RNN with one hidden layer and used a learning rate= 0.002. Due to using SGD, our model started to give NaNs. When we decreased the learning rate to 0.0002, we better results.

However, using 0.0002 with the LSTM model gave very low accuracies on both training (0.24) and validation (0.2). This was because with the LSTM model, small learning rate was not converging fast within the epoch. We found that lr=0.004 for our model was the best one.

Hidden Layer Size Experimentation:

We used a hidden layer of size 200 and this gave a higher train accuracy and lower validation set accuracy. This is because our model overfit. We decided to use a dropout =0.1 and this increased the validation set accuracy compared to non-dropout model however, the result was still lower than hidden layer size of 128.

PART 3 Analysis:

Description of our model:

Our model consists of a RNN network with one input,hidden layer. We then have 2 hidden layers one after the another and one output layer. The 2nd hidden layer is used as the input hidden layer for the next character.

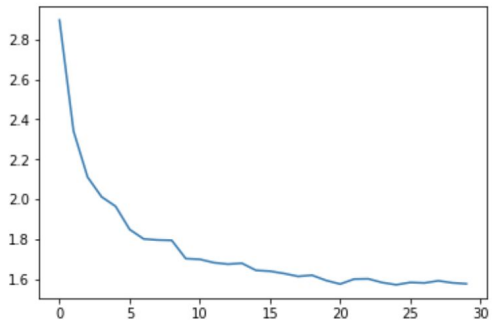
Our experimentation:

Training:

Training with random text of size 200 and 3000 epochs:

We have trained the model on Trump Speeches for 3000 epochs. Our training set for each iteration consists of getting a text from Trump speeches of size 200 randomly(We randomly choose a start index and then get the next 200 characters from that randomly choosen start index) .

We have choosen Trump speeches so that we can contrast and compare it with how different it is to Shakespeare corpora. The graph below shows the avg.loss for every 100 epochs. We can see that the training loss decreases as our epochs increase.



Text generated at different epochs:

At 100th epoch:

1. At 100th epoch , the classifier just started training and doesnt predict the text well. Clearly, the words appear to be irregular and don't make any sense.

```
[0m 17s (100 3%) 2.3634]
Wh Af fatrigh momletis. To bous they hamer in dealer hepereas. It beat theravis in bot eths a therof c
```

2. At 1500th epoch:

```
[3m 51s (1500 50%) 1.5847]
Whe people a lot of there sully while induder when you know, urder of that Iad going to be going to it
```

Halfway down the training, we can see that the words start making more sense. Words like people,a,lot,of,sully,while are seen which are more closer to english and also more closer to Trump's actual speeches. However, there are words like Whe, Induder, urder that still don't make any sense.

3. At 2300th epoch:

```
[5m 51s (2300 76%) 1.9103]
Which all of 48 people. And Polluce of if Hillinton the sceophed speciang it would country were no lik
```

At ~75% training completion, we can see that most of the words start making sense and the model started generating speeches that have more contextual meaning it to it as well. For example, which all of 48 people is a valid english sentence and the model was successful in generating it.

4. At 3000th epoch:

```
[7m 44s (3000 100%) 2.0000]
When you lobody. So they veted the forating over our millions as with that I taking the wall their the
```

At 100% training completion, the model is generating more meaningful sentences close to the english language. Training for more epochs might give better sentences but due to time of training, we have limited it to 3000 epochs for now.

Training with random text of size 1000 and 7000 epochs:

We have observed significant improvement when we use 1000 as random text size instead of size as 200. Even at 50% training , the words and sentences makes much more sense than training on 3000 epochs and on 200 random text size. As we can see from the text below, the results are quite better than the previous model and makes more sense too.

```
[45m 52s (3600 51%) 1.3473]
Wht watch. The bad hrate. I said "Try other policy. You know that knows then the made there to Hillary

[47m 7s (3700 52%) 1.0913]
Wher China will be real trement it by hire her is the tough Second problem out in the Middle AND TO MA

[48m 22s (3800 54%) 1.2432]
Whands in Party said, "When happend in a killing to spent today.  Thatas me of probably in the preside
```

Text Generated Analysis:

Where does it do well?

The text generation is doing pretty well for closed-word forms. This is because they are mostly consistent across different random samples. So, Words like Who , We , of, the, There, as , or ,your and similar words are predicted very well by the model.

Where is it going wrong?

However, the model doesn't predict the open words very well compared to closed words. For example, one of the generated text of the training is 'Who? We mean of the wordnaties the think'. Clearly, the word wordnaties doesn't belong to closed words and it wasn't generated very accurately. Taking another example, 'Whopan sign ups citity. There as great refurt of a but yous want want it' we can see that words like Whopan , citity don't exist but the model generated it. This can be accounted as open words , although they occur with great frequency, the exact words like city might not occur with such a great frequency compared to closed words and hence our model didn't predict city properly.

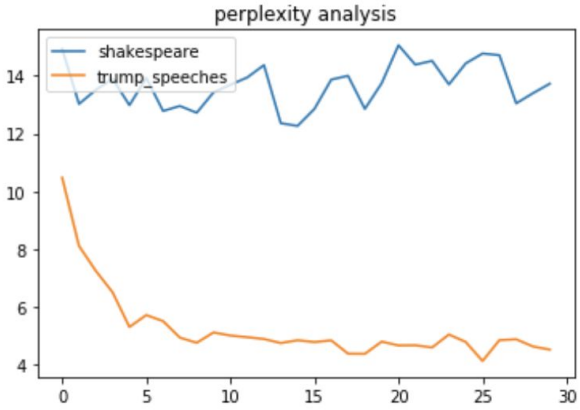
Perplexity Analysis:

a) **On one single set of random texts from Trump's speeches and Shakespeare's text:**

We randomly get a text size of 200 from both Trump Speeches and Shakespeare corpus for our perplexity Analysis. The Image below shows the corpora text obtained for both Shakespeare speeches and Trump speeches.

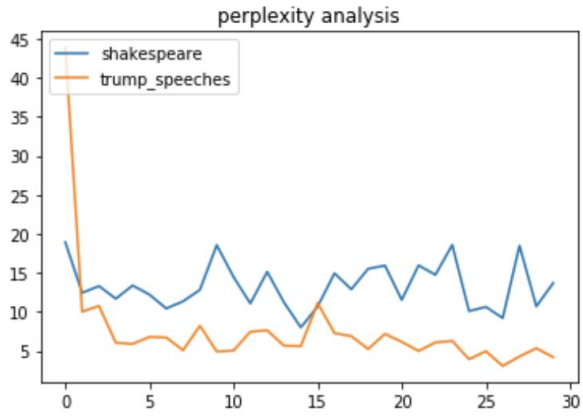
```
 Trump Speeches Random Text:  dy else even knows about it. The other candidates, most of them donat even know about i
t. Itas not their wheelhouse. Okay? Itas not their thing. Itas not their thing.
Their thing is getting re-elected.
 Shakespeare Random Text:  te,
Should presently extirpate me and mine
Out of the dukedom and confer fair Milan
With all the honours on my brother: whereon,
A treacherous army levied, one midnight
Fated to the purpose did Antonio
```

The graph below shows how the perplexity on the above mentioned texts differed after every 100 epochs till 3000 epochs. We can clearly see that perplexity of Trump's speech goes down as epochs increase and perplexity of Shakespeare's random text goes up as epochs increase. Which means that the model is learning Trump's speeches better and better with increasing epochs.



b) On random texts from Trump's speeches and Shakespeare's text generated after every 100 epochs of training for 3000 epochs:

Clearly, calculating perplexity on just one set of random texts of size 200 might have some unobserved bias in the characters that we didn't account for. Moreover, a random chunk of size 200 might not reflect the actual distribution of the corpus. To address this problem, we have also compared how the perplexity changes between these two corpus by taking random texts from both Shakespeare, Trump's speeches at every 100 epochs (till 3000 epochs) instead of taking a set of random texts before starting the training. Our results are as follows:

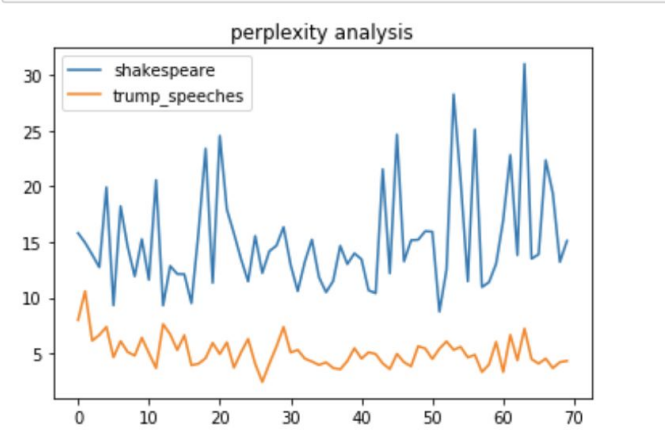


Clearly, we can see that despite taking a random string to calculate perplexity at every 100 epochs, the perplexity is consistent and has lower perplexity for Trump's speeches than Shakespeare's texts.

c) Perplexity trained on 7000 epochs and 200 random text size:

As we can see, the Perplexity for Shakespeare is high compared to Trump's speeches as we trained it for 7000 epochs. The graph shows the perplexity calculated for every 100 epochs by taking a string of size 200

from both the corpuses with a random start index. As number of epochs increase, we can see that the Shakespeare has higher perplexity than Trump's speeches and it gets more evident with higher epoch numbers. The results produced are much better than the results produced in part a,b.



Text Samples generated:

The following are the text samples generated with 7000 epochs and 1000 random text size:

1. a the way the large to be the place. Theyare being that we have to leaving it and heas not we have a lot of the time. Weare going to be ever believe that has to the thing. I want to things when the wo
2. much modement and they were leaving the people. We want to think the place and the end of the world the same that they want to be that was president. I think they was a more deals who was really on th
3. it was a fact that are strentive the record. I think itas a money. I want to happen at the fight that. I want to come that we have to be that the place that this country and the world. I was traint an

As we can see, It's quite impressive that the model learned the speeches very well and as we read these generated speeches, they seem to follow proper english conventions too.

N-Gram Comparision Analysis:

We have trained the Best N-Gram that we had last time with n= 3 and smoothing = 2 on Trump's speeches so that we can compare the perplexity with rnn models. The results are mentioned in the below table:

S.no	ngram perplexity	rnn perplexity
0	7.193475648362418	3.022588360996369
1	8.042989545446975	3.2315135469666494
2	7.466775126014917	3.08204602678741
3	7.498856870814196	3.330536178098999
4	7.955463147623153	3.419775756165892
5	7.672524155741334	3.086160690958836
6	8.197491555408563	4.017996841688867
7	7.4256994221327135	2.990797514636822
8	9.023885336276393	3.860857611504969
9	7.459874916132787	3.2427018627931323
10	8.26082834215045	3.493081551816885
11	7.647068830659189	3.305762632618756
12	9.649539328537236	4.61668952134417
13	8.047111439197536	3.7716684156056837
14	8.438144753621927	3.4387614789688494
15	7.843303407191964	3.456471587275703
16	9.93909857360584	4.0819624498686125
17	8.250518536869205	3.0111907271630742
18	8.181455717334293	3.5732631009596965
19	9.669030481880174	5.14951532748083
avg:	8.193156756750065	3.55916705918501

As we can see, the average perplexity on 20 random samples of size 1000 is 4 incase of rnn where as it is 8 incase of our best trained Ngram model. Clearly, this shows that rnn architecture generates texts better than ngrams text and are far more superior to n gram models. Our intuition for better prediction,generation for rnn’s is that just like how convolution networks that learns context for images, rnn’s can do a better training as hidden layers can activate different neurons based on the contexts of the line while training as oppossed to n-gram model which just learns the ngrams.