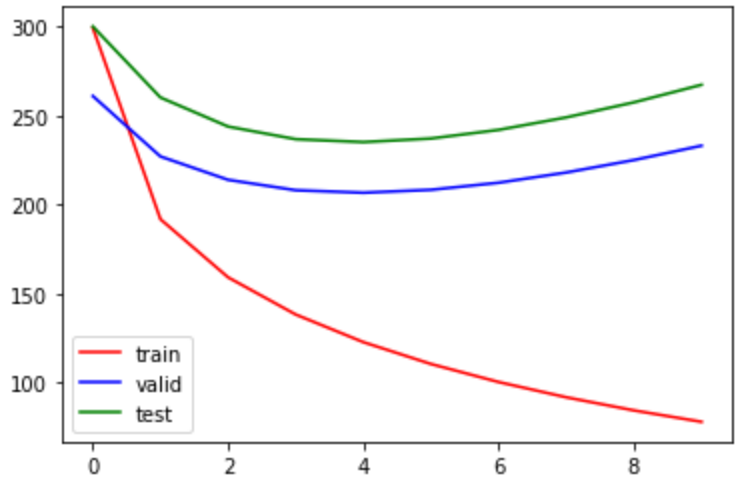
Name: Nikita Raut **NetID: NBR1932**
Name: Anupam Tripathi **NetID: AAT1666**
Name: Jiuqi Xian **NetID: JXY9577**
Name: Akash **NetID: ABS3211**
Name: Akbar Imamov **NetID: AIE6401**

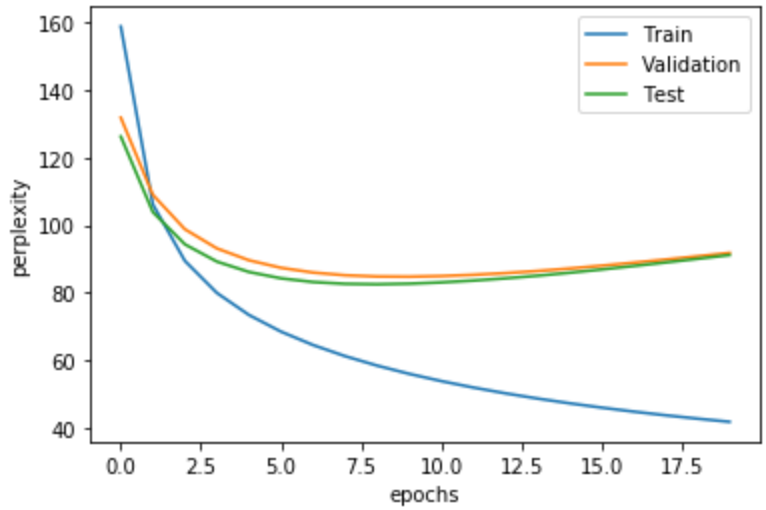
Group 4: Project 1
Topic: Comedy Movies and Drama Movies

Step 2



Step 3

Perplexity Graphs and Final perplexities
For window size n = 5

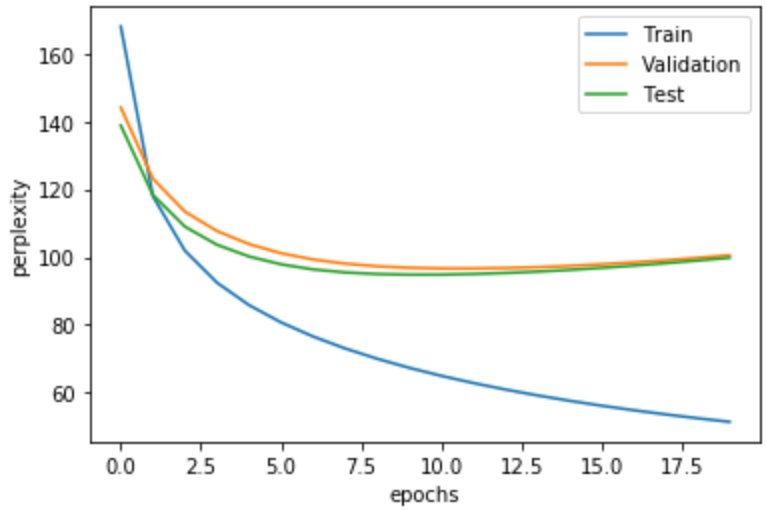


Table

Data	Perplexity
Train	41.708504571955075
Valid	91.77652842528992
Test	91.14475313636456

For window size n = 2

Graph



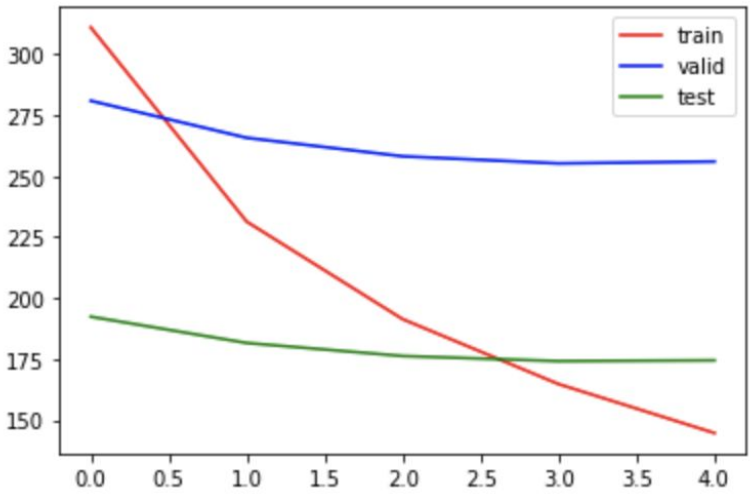
Table

Data	Perplexity
Train	51.32944023988145
Test	100.51696500221539
Valid	99.77825377716952

Comment: We observed that the model performs better for window size of 5 than 2.

Step 4
For the Hyper-parameter tuning, we used
Train: 200,000
Valid: 2,000
Test: 2,000

4.1: Tuning the learning rate

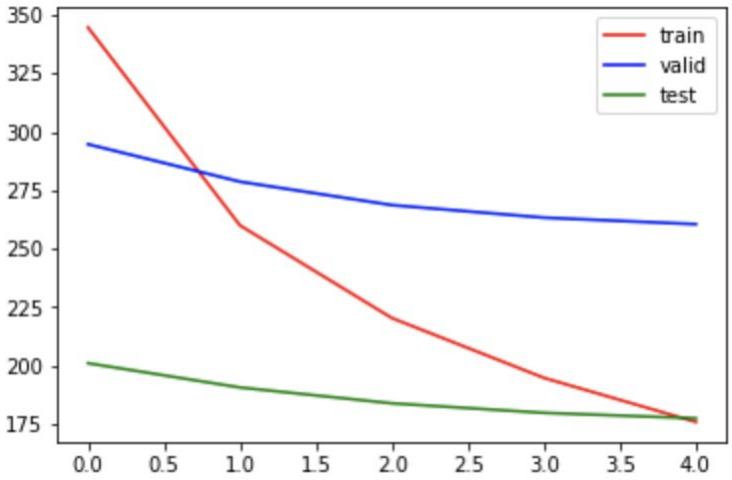



```
learning rate=0.01
n_epochs=5
window_size=5
```

n_dimensions=100

n_batches=30

r=0.1



learning rate=0.005

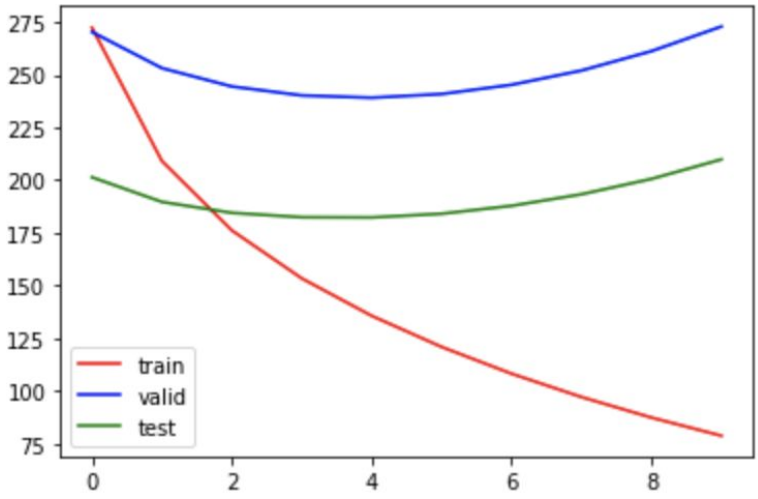
n_epochs=5

window_size=5

n_dimensions=100

n_batches=30

r=0.1



learning rate=0.01

n_epochs=10

window_size=5

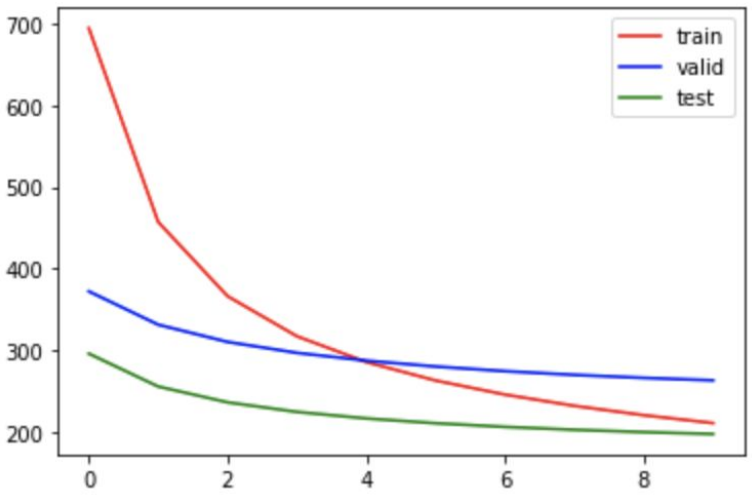
n_dimensions=100

n_batches=10

r=0.1

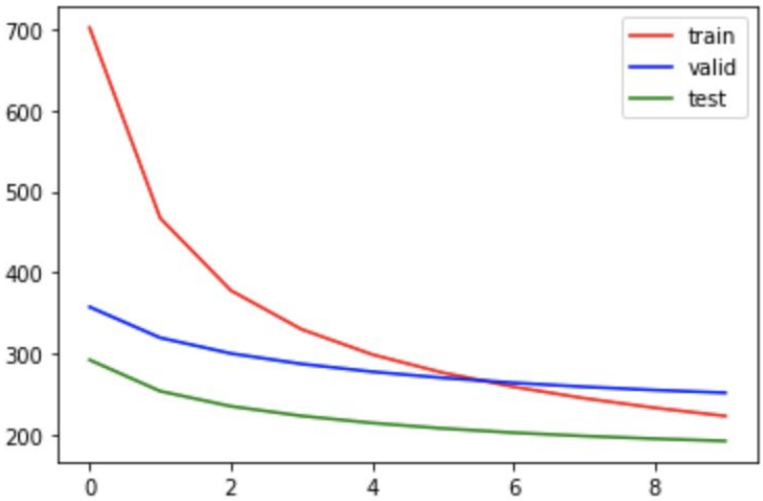
4.2: Tuning batch size

4.2.1 n_batches=50



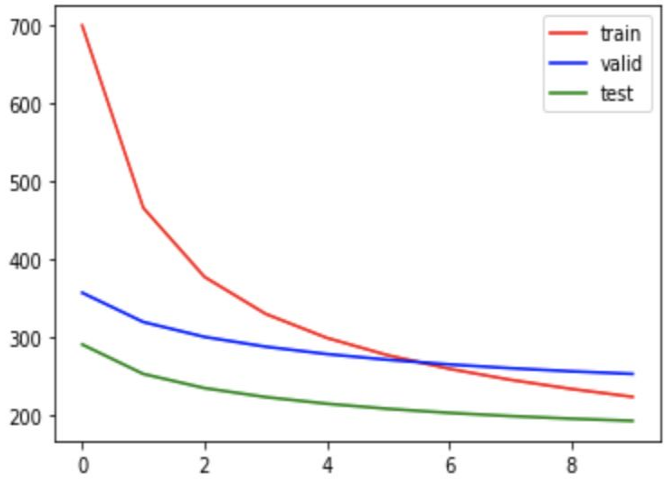
learning rate=0.001
n_epochs=10
window_size=5
n_dimensions=100
n_batches=50
r=0.1

4.2.2 n_batches=100



learning rate=0.001
n_epochs=10
window_size=5
n_dimensions=100
n_batches=100
r=0.1

4.2.3 n_batches=30




```
learning rate=0.003
n_epochs=5
window_size=5
n_dimensions=100
n_batches=30
r=0.1
```

Explanation of tuning and results: For our model, the tuning of parameters has been done after identification from parameter class and variation of these values has been charted above. The third plot is with values taken in the baseline taken in step3, (which was trained on the untagged corpus).

In contrast to this plot, we observe that the learning rate produces more variation in train curve, and by setting the `n_epochs` value has shown a pronounced change in the learning curve. Other plots with increased batch size have also been generated.

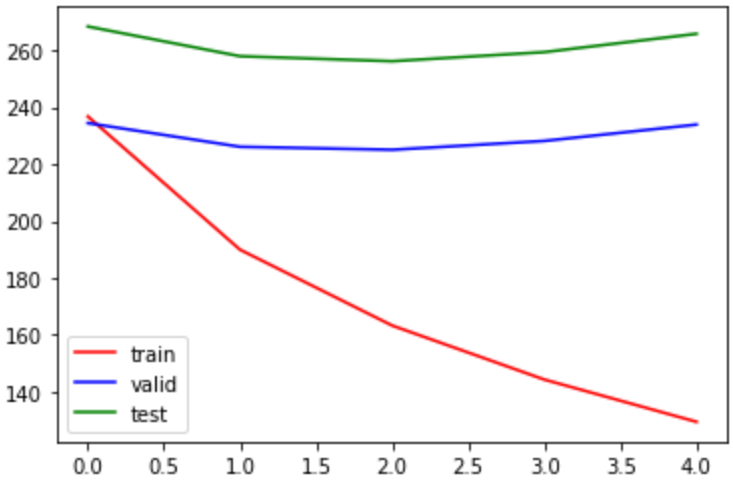
Step 5

Extra hidden layer

(Added extra hidden layer of size 50). Performance is roughly the same

Learning rate: 0.005

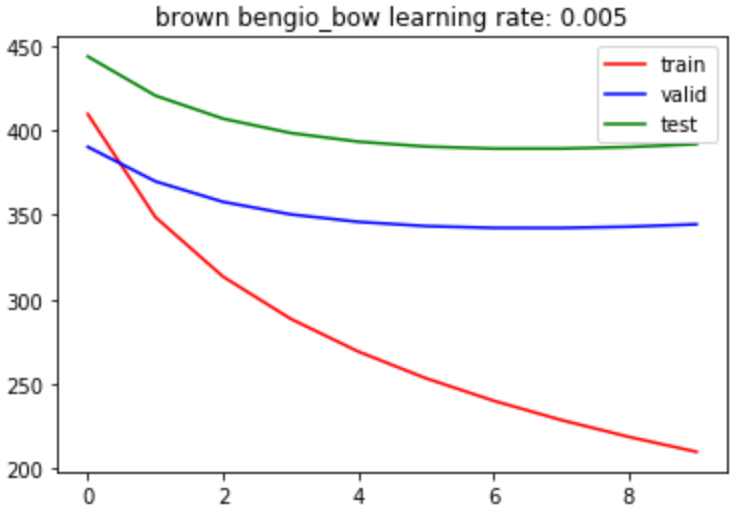
```
# y = sigmoid(tanh(b1+ Hx)*H2+b2)*H3 + Wx +b3
```



Bag of words

Learning rate: 0.005

Sum input word vectors instead of concatenating. The validation and testing accuracy is significantly higher than the original model



Brief write-up:

Step1:
In this step, we created a function to create batches. The function took in the corpus and the no. of batches and returned the batched corpus. We performed padding before creating batches.

Step2:
In this step, we implemented the Bengio-style feed-forward neural network language model with a window size of 5. We used the brown corpus for this step. The graphs for this network can be seen above. We achieved better perplexities as compared to the original paper since we used the tags from project 1.

Step3:
In this step, we implemented the model form step 2 for our wiki data corpus. We took the first 5M tokens of the 20M corpus for training and the entire test and valid set for testing and validation. We removed the direct connection from the network and trained the model for a window size of 2 and 5. As expected, the model performed better for a window size of 5. The perplexities went as low 41 for the 20 epochs we trained on.

Step4:
Explanation of tuning and results: For our model, the tuning of parameters has been done after identification from parameter class and variation of these values has been charted above. The third plot is with values taken in the baseline taken in step3, (which was trained on the untagged corpus). In contrast to this plot, we observe that the learning rate produces more variation in train curve, and by setting the `n_epochs` value has shown a pronounced change in the learning curve. Other plots with increased batch size have also been generated.

Step5:
For this step, we have added an extra hidden layer to observe whether it improves the perplexity of our model. Since we don't exactly know what happens in hidden layers, we wanted to observe the results of the experiment and make a conclusion. We added an extra hidden layer with 50 nodes to the standard Bengio NNLM, and we observed that the results didn't change significantly compared to our initial model. You can see the plot of train/valid/test relative to # of epochs. Therefore we concluded that it is not necessary to add a new hidden layer, which would add extra computation time. For the second experiment, we have decided to sum the input word vectors instead of concatenating them. As a result, we have got much more accurate results for testing as well as validation data.