NLP Unit 2 Evaluation *<Team no. :* ***N16****>*

Our Model

We created two ways to test/assess our model :

* **Test 1**

Output consists of highest probable number for each number in input sequence. Thus, the output sequence has many repeated elements and sometimes the output has numbers not present in the input sequence. Apart from this, most of the output sequences are in sorted order by themselves.

*Eg:-* For Input [2,1,4,7,3] we obtain something like [1,1,3,3,7]

* **Test 2**

In this we take numbers only present in an input sequence to be arranged in output sequence. Also we take care of duplication by moving to next highest probability if the highest probable number is already a part of output sequence. Hence we eliminated the repetition of numbers in the output sequence and also eliminated the presence of numbers which were not in the original input sequence from the output sequence .

*Eg:-* For Input [2,1,4,7,3] we obtain something like [2,1,3,4,7]

We varied the following hyperparameters to analyse the effects on our models

* Dataset Size
* Hidden Units
* Learning Rate
* Input Range

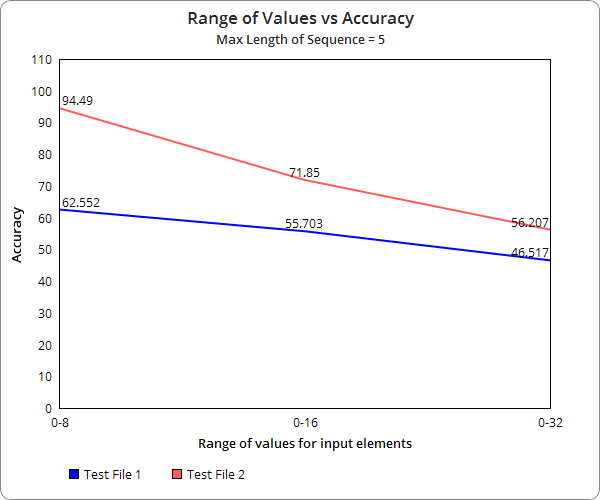
*NOTE : Accuracy along y-axis in all graphs is in %.*

* Varying the Input Range

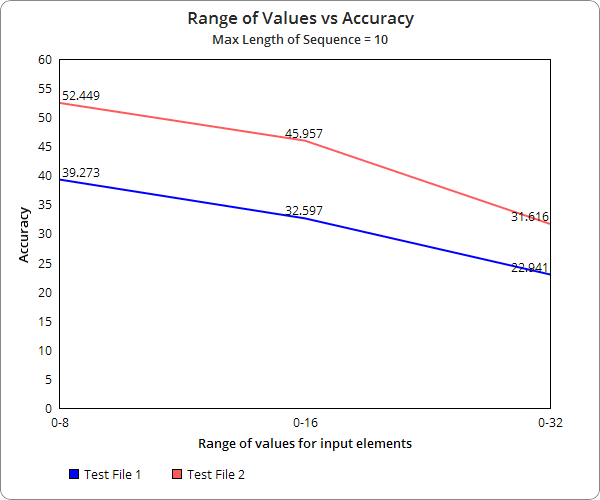
As we were trying to understand factors affecting accuracy, we tried to get our hands dirty by changing the input range. From our analysis we found out that the range of values passed as input and also the length of each sequence to be sorted affect the accuracy of our model .

We achieved the following results :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Max sequence length** | **Range of Input Elements** | **Data Set** | **Test1 Accuracy** | **Test2 Accuracy** |
| 5 | 0-8 | 10000 | 62.552 | 94.49 |
| 5 | 0-16 | 10000 | 55.703 | 71.85 |
| 5 | 0-32 | 10000 | 46.517 | 56.207 |
| 10 | 0-8 | 10000 | 39.273 | 52.449 |
| 10 | 0-16 | 10000 | 32.597 | 45.957 |
| 10 | 0-32 | 10000 | 22.941 | 31.616 |

.

*We achieved the above results when we fixed the maximum length of each sequence to be* ***5***



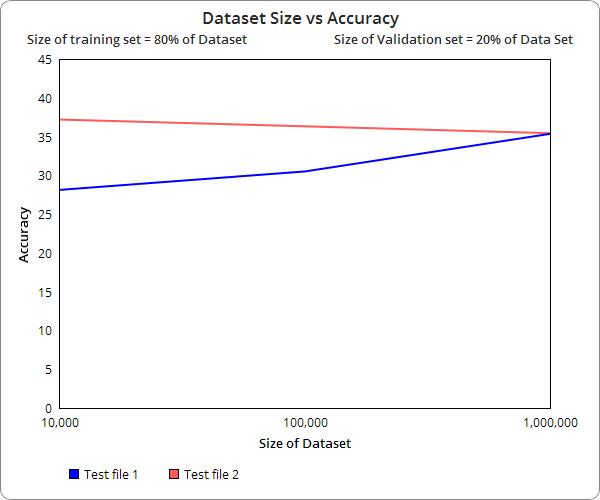
*We achieved the following results when we fixed the maximum length of each sequence to be 10.*

From the above two graphs, we see that the accuracy increases as we decrease the range of input values in the data set . The accuracy also increases significantly as we decrease the length of each sequence in our input.

***Inference*** *-* We believe that this increase in accuracy with decrease in range of input values is because our RNN gets trained consecutively on a much smaller set of values many more times , hence causing it to learn repeatedly on the same set of inputs.

* Varying the Size of Dataset

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Dataset (N)** | **Training Sample** | **Test Sample** | **Accuracy (Test 1)** | **Accuracy (Test 2)** |
| 10,000 | 8,000 | 2,000 | 28.122 | 37.193 |
| 100,000 | 80,000 | 20,000 | 30.502 | 36.32 |
| 1,000,000 | 800,000 | 200,000 | 35.357 | 35.43 |



Following were our observations

* Test File 1

The accuracy increased, with increase in the Dataset size. We see that accuracy increased by ~2.4%, for increase in 10X the size of Dataset and ~7%, for increase in the 100X size the Dataset.

*Note : Size of original dataset was 10000*

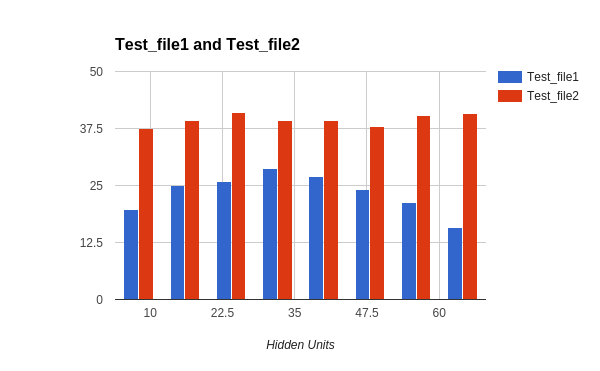
* Test File 2

Here we observed that, it stayed almost the same (Delta being ~2%) irrespective of increase in size.

* Varying the Hidden units

*Note: The following observations were made with dataset size - 10000.*

|  |  |  |
| --- | --- | --- |
| **Hidden Units** | **Accuracy(Test\_file1)** | **Accuracy(Test\_file2)** |
| 8 | 19.7 | 37.5 |
| 16 | 24.9 | 39.2 |
| 24 | 25.9 | 40.9 |
| 32 | 28.7 | 39.2 |
| 40 | 26.9 | 39.3 |
| 48 | 24.2 | 38 |
| 56 | 21.2 | 40.3 |
| 64 | 15.7 | 40.8 |



Following were our observations

* Test File 1

On increasing the hidden units in steps of 8, there is an increment in accuracy till the hidden units reach 32, after which there is a steep descent in accuracy on further increasing the units.

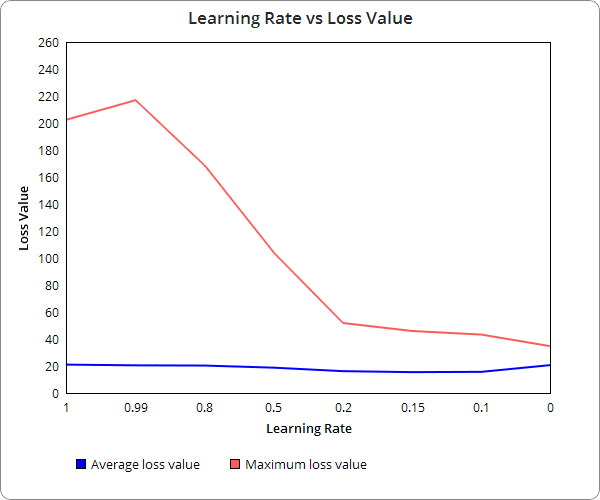
***Inference*** *- The reason behind this maybe underfitting before hidden units = 32 and overfitting after that.*

* Test File 2

The accuracy for this remains more or less same independent of an increase in number of hidden units.

* Varying the Learning Rate

|  |  |  |
| --- | --- | --- |
| **Learning Rate** | **Average loss value** | **Maximum loss value** |
| 1 | 21.021 | 202.451 |
| 0.99 | 20.492 | 216.902 |
| 0.8 | 20.328 | 168.482 |
| 0.5 | 18.757 | 103.855 |
| 0.2 | 16.177 | 51.831 |
| 0.15 | 15.453 | 45.923 |
| 0.1 | 15.674 | 43.286 |
| 0 | 20.668 | 34.661 |



* Variation in the learning rate of the neural network results in changes in the loss value for each subsequence of data. The ideal range of learning rate values for most applications of Bidirectional RNNs are 1.0 - 0.01.

As a rule of thumb, our approach towards converging to an appropriate value of the learning rate was to -

1. Start with a high value for the learning rate, for instance 0.5.
2. Decrease values by an order of learning rate/2 as the loss values start flattening out.
3. Continue the procedure until an optimum value of the learning rate is obtained.

***Inference*** *: Increasing the learning rate will decrease the accuracy, which in turn causes* ***loss value to increase*** *[because there is chance of overshooting the target and needs backtracking].*

* What went Right?

As per our observations :

* Although the output sequence was not same as input sequence in many cases and it also had a lot of repetitions but most of the sequences in itself were sorted.
* Also variations in hidden units VS accuracy illustrated the concept of under and overfitting.
* With larger values in the training data, the accuracy of the model increases as is expected.
* Shortcomings
* Many of the output sequences have repeated numbers .
* The output looks more like a series of sorted chunks rather than a single sorted sequence [There were many sequences in output which were distorted by couple of values. Eg [1,3,15, 2, 19 , 27 , 30 ] where the underlined chunks are sorted but the correctness of sequence as a whole is distorted due to the presence of ‘2’ in the middle ]
* The accuracy for the model is not up to the mark (to make it usable).
* Results
* We finally obtained an accuracy of **35%**(approx.) with an input range of **[0,32],** hidden units - **32** and learning rate of **1e-1** trained on a dataset of 1 million out of which 20% was used for testing - <*Unmodified Problem Statement*>
* After changing input range to **[0,8]** keeping all other hyperparameters same, we obtained an accuracy of **62.6%**.