Word Recognition HW

To start with, I take 7 random positive words from website: <https://www.randomlists.com/random-words?dup=false&qty=20>

Those words came out as:

**['nice', 'toothpaste', 'highfalutin', 'childlike', 'authority', 'halting', 'found']**

As seen above, they have different length, so we use windowing techniqueto represent all words at same length. How we do is by first finding the word with longest length (highfalutin = 11 letters) then adding 1, so our window length becomes 12. After this we start padding spaces to all other words with shorter length. As seen below all padded words now have length of 12:

'nice ', 'toothpaste ', 'highfalutin ' …

Then we need to create training data(size of 10,000) out of original 7 words. Training data will contain both positive words and negative words. We create itby randomly shuffling the 0-3 letters of original 7 positive words. After that, we pad the spaces to words. Note that approximately 55% of these 10,000 words inside training data are positive, the rest is negative.

Then as we have to encode the input words as binary rather than letters, next step is to represent each letter of word in binary form. We encode each letter with 5 bits, so by this way we convert input words of length 12 into length 60, i.e;

' ' = 00000 (0)

'a' = 00001 (1)

...

'z' = 11010 (26)

So, after all the above steps, we have **encoded train data** which contains 10,000 encoded words each having length of 60.

Then we create train target (correct outputs for network)**.** We encode the output as a vector of binaries where "1" will be in ith position if the model recognize input word as ith word from original positive list of 7 words. For example, if we give word “nice” to the network, the network should produce as below:

[1, 0, 0, 0, 0, 0, 0] indicating that that word is present in 0th index of original positive list of 7 words.

**Network Specifications:**

60 – 10 – 10 – 7 (60 input layer neurons, 10 first hidden layer neurons, 10 second hidden layer neurons, 7 output layer neurons)

Activation function: relu

Solver: adam

Batch size: 300

Learning Rate: 0.001

Iterations: 200

We also decode the output produced by neural network. So, if any single neuron out of 7 neurons of output layer becomes 1, that means that given input to network is recognized as word from original 7 positive words list with index same as that activated output neuron. For example, if network output is:

[0, 0, 1, 0, 0, 0, 0] then it means that network recognized input as positive word which is **'highfalutin'**(index = 2 from original 7 positive words list).

After training and testing the the network, some screenshots of results are as below:

A picture containing table

Description automatically generated

.

.

.

A picture containing text

Description automatically generated

Screenshots are taken from **results.txt file.**

As seen above, network trained 198 iterations with last loss of 0.00032190

Output, Response, Error, Accuracy and Reliability values for each negative word of test dataset are given.

At the bottom, final accuracy for all test datasets were given.

**For Negative data :** Our network could correctly classify the negative words with 99.94% accuracy (4997/5000).

**For Negative+Positive data mixture:** Our network could correctly classify all words with 100% accuracy (5000/5000).