**Unit 3 Evaluation - Group N3**

**Objective**: To build a RNN based sentiment analyzer

**Inputs**: 4 text files - 2 for training and 2 for testing.

**Training file**: one for positive reviews and the other one for negative reviews. These two files contain 1000 reviews each.

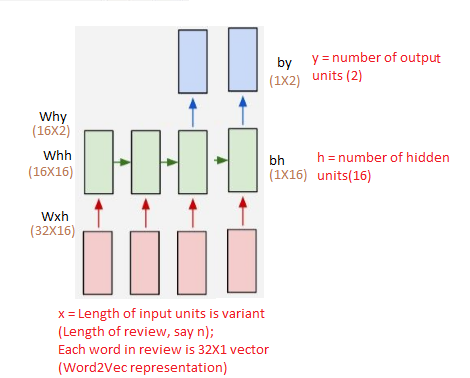
**Testing file**: one for positive reviews and the other one for negative reviews. These two files contain 100 reviews each. Each review is separated by a “\n”.

**Procedure:**

1. Getting the input:

Given input had two json files, with positive data in one and negative in the other. We extracted 1000 reviews, cleaned it(removing special characters, smileys, etc) from file1 and stored them in train\_pos\_processed.txt. Similarly, we extracted 1000 reviews from file2 and stored them in train\_neg\_processed.txt, 100 more from file1 and stored them in test\_pos\_processed.txt, 100 more from file2 and stored them in test\_neg\_processed.txt.

2. Model:



***In the figure above,***

black - The Parameters of the model used

brown - The dimensions

red - The model

***Hyperparameters:***

1) Number of units in each layer:

- Input: Length of sentence; variant;

- Word representation: 32X1 vector

- Hidden : 16 units

- Output : 2 units (First unit represents negative and second unit represents positive)

2) Learning rate: 1e-1

3) Epochs: We tried with 100, 500 and 1000

**Functions:**

Input : (32X1) vector whose size is number of sequence of words in a given review

Hidden (Forward): tanh function with (Wxh.x + Whh.h(prev) + bh) as its parameter

Output: Softmax function with (Wyh.[h(forward)]+by) as its parameter. Contains probability distribution of two classes (positive/negative)

3. Procedure

1. For training we passed positive reviews fist and negative reviews later. Each review had a set of words. Each word was converted to a 32 sized vector using gensim Word2vec model and passed as input for rnn. The target was 0 for negative review or 1 for positive reviews.

2. For each such input, forward propagation and backward propagation was done as explained in the functions above. Here, we want to generate the output only after the entire review is read, so, if the review length is t, output is generated at (t+1)th time step. Also, during backward propagation, we copy the probabilities at last time step only as we do not require it after thhat for the reason mentioned above.

3. After the weights were updated, this model was used to test data for new sequence of lists of reviews.

4. Building the model

In the forward propagation,

1. h(forward) is evaluated from left to right

2. Output, o is evaluated (we evaluated it from left to right).

Error propagation in backward propagation:

1. Errors were propagated from output to hidden using first order differential function of softmax.

2. Errors were propagated from hidden to input from right to left, using first order differential function of tanh.

3. Weights were updated.

Process continues till loss gets minimized beyond a threshold

5. Output:

train\_data = {1:train\_pos, 0:train\_neg}

test\_data = {1:test\_pos, 0:test\_neg}

The output layer is 2x1. The indices 0 in the above code snippet represents a negative tweet and 1 represents a positive tweet. We’ve labelled them as 0 and 1 in the form of a dictionary.

Softmax distributes probability among these two units. The index which has the maximum probability is returned. While testing, we used np.argmax(ps[len(inputs)-1]) which returns the index with the maximum probability.

**Results and Discussions:**

|  |  |  |  |
| --- | --- | --- | --- |
| Hidden units | Alpha | Number of epochs | Accuracy |
| 32 | 1e-1 | 10 | 48.3 |
| 32 | 1e-1 | 25 | 53.33 |
| 32 | 0.1 | 25 | 50.00 |
| 32 | 0.001 | 25 | 49.90 |
| 64 | 1e-1 | 25 | 48.00 |
| 16 | 1e-1 | 25 | 50.00 |

All the below mentioned points is what we arrived at initially:

1. As we trained the model with positives first and negatives later, the output was more biased towards negative reviews.

2. By increasing the number of epochs, the accuracy increased by small amount.

3. Changing learning rate (Alpha) values, did not affect the accuracy much.

4. Increasing the number of hidden units, improves the accuracy by a small amount.

**Improvising:**

As tweaking the hyper parameters did not yield good results, we increased the number of training and test cases. Also, we passed the reviews alternatively (pos-neg-pos-neg-...-pos-neg) to get rid of the model being biased to one type of output. This improved the accuracy to 70%.

1. The reason behind this could be addition of more vocabulary

2. The test data might have had words as in the train data

3. Number of iterations in an epoch increases with increase in data, leads to better learning of parameters.

4. Model is less biased

*What we could have tried?*

1. Implementing LSTM/GRU as hidden units to save only necessary data from previously collected data and clear the data which is not of much importance.

2. Instead of binary model, we could have tested the same for multi-class model, usually the ratings are from 1 to 5 in real world scenario.