# Natural Language Processing:

# Assignment 2 - Report

**Name:** Payal Mehta Date: 25th Oct 2019

**SBU id:** 112715351

==================================================================

## 1. Model Implementation

### a. Deep Averaging Network (DAN):

I have implemented DAN in three simple steps as described below:

1. Input to first layer:
   - The input to the first layer is the vector average of the embeddings for every token in the input sequence. To calculate the vector average, I have taken the product of the input vector sequence and the sequence mask.
   - The sequence mask is a combination of 1s and 0s. Entries with 1 indicate that token is a real token, and 0 indicate that it's a padding token. I have applied a random uniform distribution where the generated values follow a uniform distribution in the range given. I have given the range as the dimensions of the sequence mask.
   - The probabilities are then picked based on whether they are greater than dropout (DAN works better with a regularizer like dropout where we randomly drop token embeddings based on dropout). I have multiplied these probabilities with the sequence mask, which in turn is multiplied with the input vector sequence to get our final set of word embeddings. To calculate the average, I have then divided this final product with the actual number of word embeddings. This average is the input to the first layer.
2. Subsequent layers:
   - The output of each layer is then fed to the next layer.
3. Output:
   - The combined vector is returned as the output of the final layer. Layer representations are returned as the output of each layers.

Handling of NaNs:

When the dropout probability is so large, that after dropping out we are left with 0 words in the sentence, the denominator in the average becomes 0, resulting in NaN loss. To handle this, I have checked when the denominator becomes 0, I divide by 1 instead. Doing this does not affect our average as anyway since there were no real words in the sentence, the numerator was 0.

When the input sequence is a bigram, we avoid applying dropouts and thus avoid probability of NaN losses.

**b. Gated Recurrent Unit (GRU):**

The gated recurrent unit is an improvement to the Recurrent Neural Network (RNN) as it helps solve the problem of vanishing gradients. This is done by using update gates which tells the model how much of the past information needs to be passed to the next layer and the reset gate tells the model how much of information from previous layers to forget.

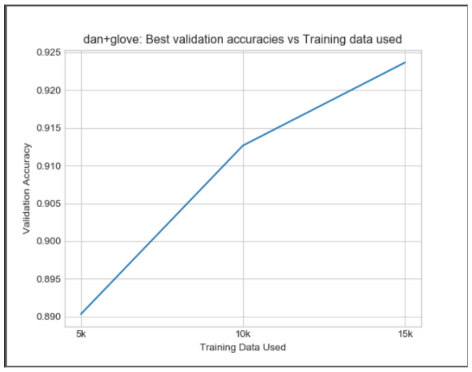
I have implemented it as below:
The input to the first layer is simply the vector sequence provided along with the sequence mask. At every step, the output obtained is passed to the subsequent layers. The last layer's output is returned as the final output along with the layer representations as individual output of all layers.

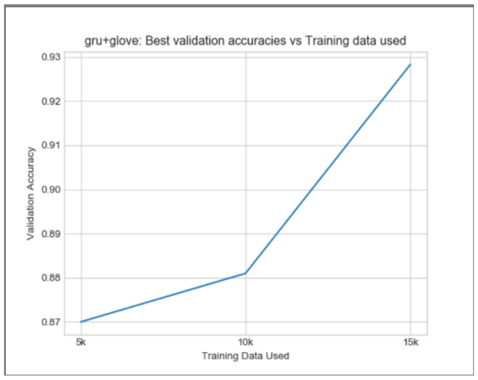
**c. Probing Model:**
I have used softmax as the activation function on the dense layer and then passed inputs to the pretrained model. Logits are calculated by applying the layer representations and the layer number to the above obtained output. Logits are returned as the output.

## 2. Analysis: Learning curves and Error Analysis

Increasing the training data:
Observations:

- As we increase the size of the training data, we see that the Deep Averaging Network (DAN) model learns faster at lower data size of 5k and 10k and becomes stagnant as we increase the data size further.
- The Gated Recurrent Unit (GRU) performs better at larger data sizes (15k) but has a steep learning curve (i.e. takes longer to learn), but once it does it performs better and gives higher accuracy.

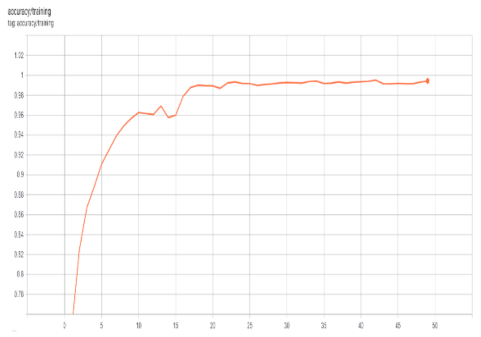


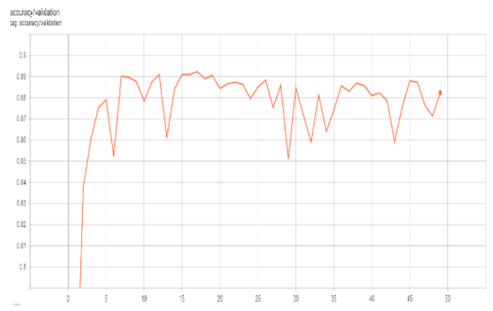


Increasing the training time:

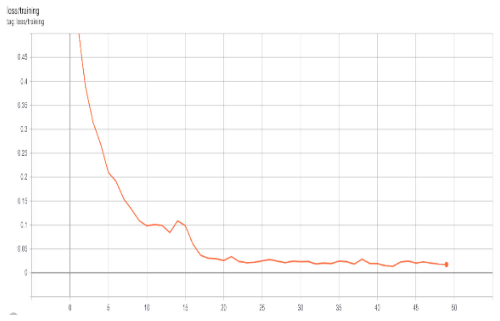
Observations:

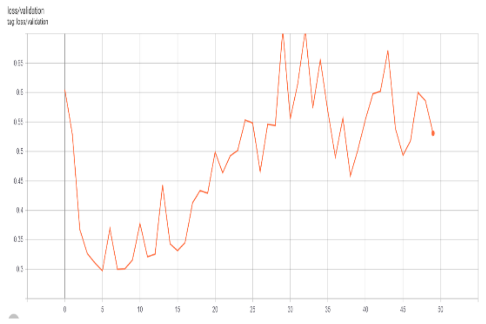
- Whenever a model is trained for a longer training time, it begins to perform better on the training dataset as it has seen the data for a long time. This phenomenon is known as overfitting.
- Accuracy of the DAN model increases as we increase the number of epochs aka the training time.
- However, this leads to overfitting issues and ends up performing worse on the validation/testing data sets because of this.





- In contrast, because of training the model for more number of epochs, the losses are lower for training data sets and higher when validation data set is run because of overfitting.





**Error Analysis:**

Advantages:

1. Advantages of GRU over DAN
   - GRU works better for larger datasets when compared to DAN.
   - GRU takes into account the sequence of words and hence performs better for the bigram order task.
   - GRU uses update and reset gates which allows them to store and filter information. That eliminates the vanishing gradient problem since the model has the configuration to take relevant information from the input and pass it to the next layers. This is not possible in DAN.

2. Advantages of DAN over GRU
   - DAN learns faster – takes considerably lesser amount of time in training the model.
   - It performs incredibly better for smaller datasets as compared to GRU.

<u>Failure Cases:</u>

**Sentence1: -** "The food was delicious."

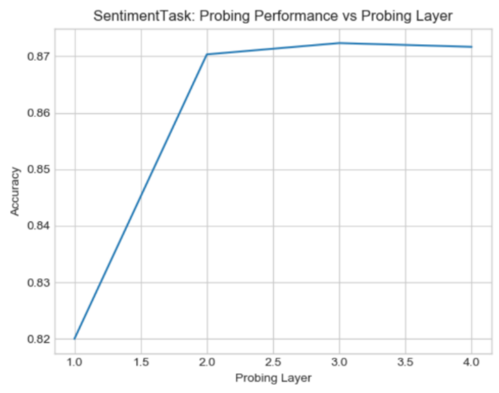
DAN identifies this as a positive sentiment, while GRU fails at this and identifies this as a negative one because it does not train effectively because of the higher number of parameters and a lower training dataset (in this case – 5k)

**Sentence2: -** "The movie was not awesome"

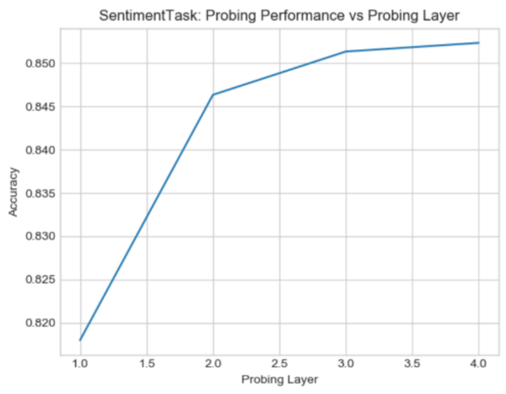
When this sentence was fed to the models which were trained on 15k data, GRU marked this as a negative sentiment outputting a 0 and DAN predicted it as positive. This behavior is seen because the deep averaging network (DAN) does not take into account the relative order of words in a sentence. As it sees a positive word ("awesome" in this case), it assigns a very high weight to it and thus marks the sentence as a positive sentiment – which is incorrect.

## 3. Probing Tasks: plot and observations/explanations

### a) Probing Sentence Representation for Sentiment Task



**DAN**



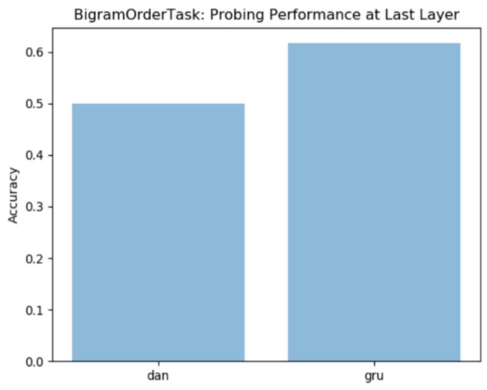
**GRU**

- As per above figure, we can see that for the sentiment analysis task, DAN performs better at layer 2 and then sees a dip as we increase the layers above 3.
- GRU doesn't perform as good as DAN however starts performing better as we increase the layers and will eventually outperform DAN.
- As we increase the number of layers, the differences between the words go on increasing and after a certain point, when the number of layers is too high, the differences no longer represent correctness, hence the models perform better at a lesser number of layers.

The accuracies for DAN at various layers are:
[0.82, 0.8703333333333333, 0.8723333333333333, 0.8716666666666667],

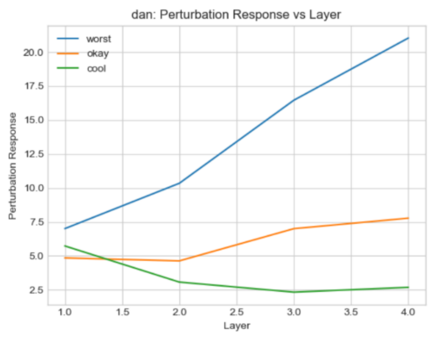
The accuracies for GRU at various layers are: [0.818, 0.8463333333333334, 0.8513333333333334, 0.8523333333333334]

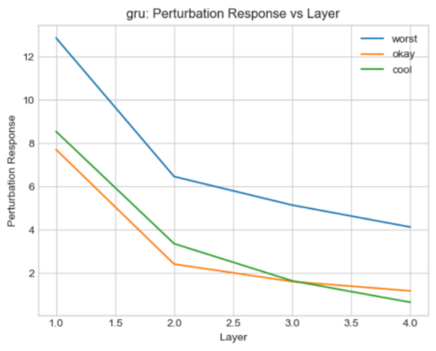
## b) Probing Sentence Representations for Bigram Order Task



- The Gated Recurrent Unit (GRU) performs better on the bigram task as it records the order in which words appear. DAN will never understand in which order two words appear as it simply picks words, averages them and passes to the feed forward layers.
- GRU, on the other hand is a gated unit and at each step it uses past information from previous layers to remember the sequence of words and thus results in better prediction.

## c) Analyzing Perturbation Response of Representations





- In the first layer itself, DAN correctly identifies the weight of each word and the difference with the base word. GRU on the other hand identifies this and moves “cool” closer to base word “awesome” only after layer 3.
- However, with the increase in number of layers, DAN keeps increasing the differences between the base word (word “awesome” in this case) and other words – “worst”, “cool” and “okay”.

- It begins with a “real” difference but goes on magnifying differences until it no longer maintains it. The differences between negative and positive sentences/words becomes increasingly amplified with the increase in number of layers.
- The gated recurrent unit on the other hand, maintains a relative distance as it takes into account the sequence of words and finally lowers the gap between “awesome” and “cool” with the increase of layers.

**4. References:**

**1.** https://www.ijcai.org/proceedings/2019/0705.pdf
**2.** https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be
3. https://people.cs.umass.edu/~miyyer/pubs/2015_acl_dan.pdf