CSE538: Natural Language Processing **(Due: October 25th)**

Submission Homework #2

Instructor: Niranjan Balasubramanian *Name:* Amanpreet Singh *Student Id:* 112044129

1 Model Implementation

I) **Deep Averaging Network:**
Deep Averaging Networks(DAN) have been implemented via a multi-layered Feed Forward Neural Net model. The number of layers are supplied to the 'sequence_to_vector' and by using the keras.layers.Dense class in Tensor Flow, those many layers are then created and maintained in a list to be iterated over.ReLU is used as the activation function for each layer.
Masking was an important aspect of the implementation. To have a constant number of embeddings for each sentence in the batch, shorter sentences were padded which had to be converted to 0's by multiplying the sentence embedding with the sequence mask which acts as an indicator of legit token and padding words.
The second and most distinguishing feature was implementing the dropout regularization for training. To emulate Bernoulli Trials over words of the sentences, a random uniform distribution was first created to generate probabilities between 0 and 1. Then, the values less than the dropout factor provided to the model were converted to 0 according to [1]. Uniform distribution was chosen as by default it provides values between 0 and 1. This was then multiplied to the padding masked vector sequence to get the final masked input which is then reduced mean to get the average and supplied to the first layer of the Feed Forward Network. The outputs of each layer are then fed as inputs to the subsequent layers and layer representations are obtained. The output from the last layer and layer representations are returned.

II) **Gated Recurrent Unit:**
For Gated Recurrent Units (GRU), specialized GRU encoders are provided within Tensor Flow as keras.layers.GRU. As before, multiple GRU encoders were used for implementing the various layers. The activation function used here is tanh. Unlike DANs, the sequence mask is provided directly as an input alongwith the original vector sequence to the GRU network and all the optimization then happens under the hood. Again, for each layer of the network the output sequence becomes the input for the next layer. As a sequential model, the layer representation at every level represents the sentence embeddings generated till that level.

III) **Probing Model:**
The probing model uses the sentence representation generated at one of the middle layers in the above networks in a classification task, namely Sentiment or Bi-gram classifications. For this purpose two things are required mainly, loading a pre-trained DAN or GRU model and creating a new classifier to use the embeddings generated from the pre-trained model. A single Dense layer is created for this purpose with no activation. The pre-trained model is used for predicting input representations, from which the representations of a intermediate layer are fed to this layer to learn the weight and bias.

2 Analysis

2.1 Learning Curves:

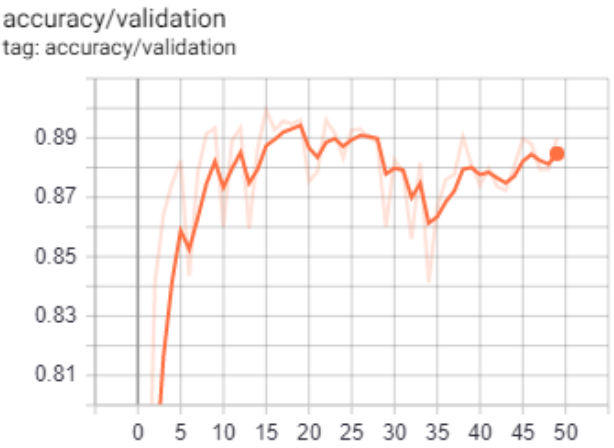
Changes with:

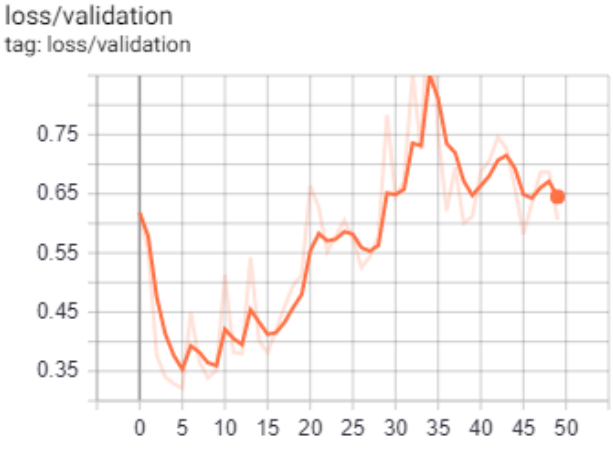
(i) Increasing the training data:

Data Size	DAN Accuracy	DAN Loss	GRU Accuracy	GRU Loss
5k	0.8823	0.3211	0.87	0.3246
10k	0.9110	0.2779	0.8760	0.3044
15k	0.9257	0.2194	0.9313	0.1951

As can be seen from the above table, increasing the data size helps to increase the validation accuracy as well as decrease the loss. This observation makes sense, as with the increasing number of training samples, the model learns better about the features of the text. The change is more steep in case of GRU as compared to DAN. Another aspect to consider is the trade off between the accuracy and the time taken to train the model. Even with a 10k increase in data size, virtually no change was observed in the DAN training time, but for GRU every epoch took 2X time at 15k as compared to 5k.

(ii) Increase training time (number of epochs): This analysis was done only for DANs till 50 epochs as GRU would have taken a long time to train for so many epochs. Coming to the observations, for the training data, both the accuracy and loss become almost constant after the 30^{th} epoch, with only nominal increase observed till the 50^{th} and eventually ending at around 0.994 accuracy and 0.018 as the loss. For the validation set though, the accuracy keeps fluctuating between 0.86 and 0.89 peaking with 0.8942 at the 19^{th} step. As for the loss, the model doesn't do much good as it starts out at 0.6185 but after the 15^{th} epoch, it increases constantly ending up at 0.606. This would indicate that beyond a certain threshold the accuracy wouldn't matter and the data would begin to overfit resulting in a rise in the actual loss when the model is tried on test data.





2.2 Error Analysis:

(i) Explain one advantage of DAN over GRU and one advantage of GRU over DAN.
DAN over GRU:
DAN models are simplistic in their nature and easy to train. This translates into shorter training time even with many non linear layers. GRU on the other hand capture more information but are excruciatingly slow to train. This was one of the precise features presented in the dropout paper for DANs which is the basis of the assignment.
GRU over DAN:
As mentioned above, DANs are not able to capture much of the contextual and individualistic information from tokens as the input is the averaged out word embeddings. GRUs perform exceedingly well to capture the sequence information as at each level, the partial representation of the text is generated and then passed on to the next.

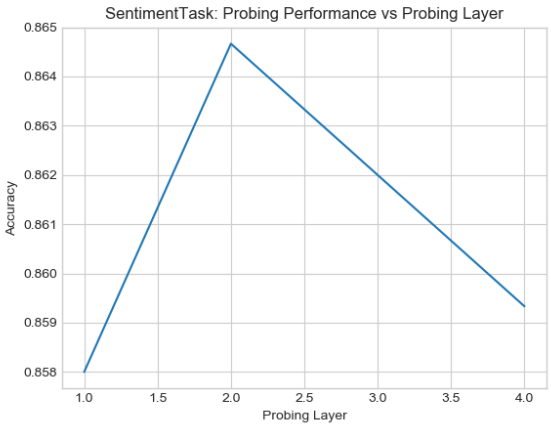
(ii) Use the above to show and explain failure cases of GRU that DAN could get right and vice-versa. Use one of your trained models to do this analysis.
DAN gets training speed right
Both the DAN and GRU models were trained on the same samples of varying sizes and same number of layers. For 5k samples, DAN took about 2 minutes to train while it took 10 minutes for the GRU model. When the size was increased to 15k, DAN models were trained within 5 minutes while it took around 15-20 mins for every step in the GRU training. Similar differences were observed when the models were trained with 1,2,3 and 4 layers of Neural Nets. Also, with increased data size, the difference in accuracy observed wasn't much for both.
GRU gets sequence right
In the Bi-gram task, the accuracy obtained for GRU is 10% more than for DAN at around 60%. As an example, the bi-gram "your ego" from test data is classified as valid by the GRU model but invalid by the DAN model. There are several similar examples throughout the dataset.

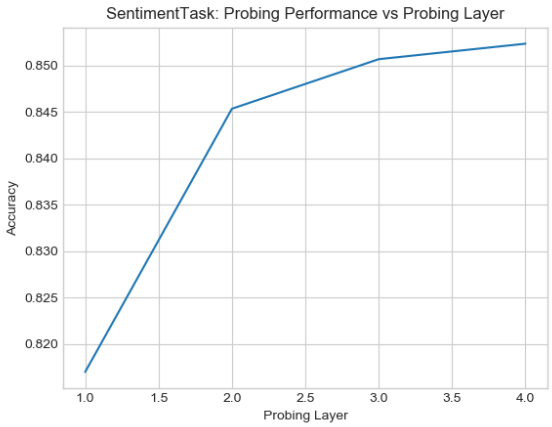
3 Probing Tasks

3.1 Probing Sentence Representation for Sentiment Task

Table 1: Validation metrics for Sentiment Analysis

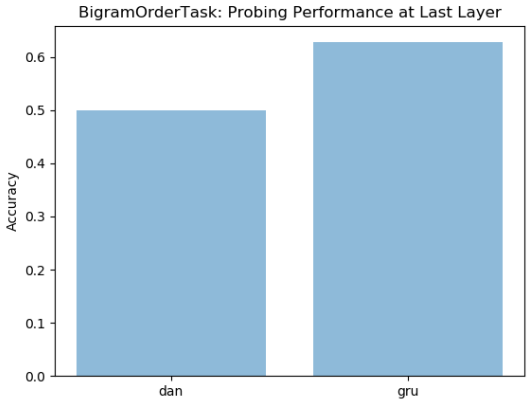
No. of Layers	DAN Accuracy	DAN Loss	GRU Accuracy	GRU Loss
1	0.8797	0.4536	0.8360	0.4063
2	0.8820	0.3064	0.8680	0.3320
3	0.8860	0.3390	0.8670	0.3155
4	0.8847	0.2968	0.8690	0.3150





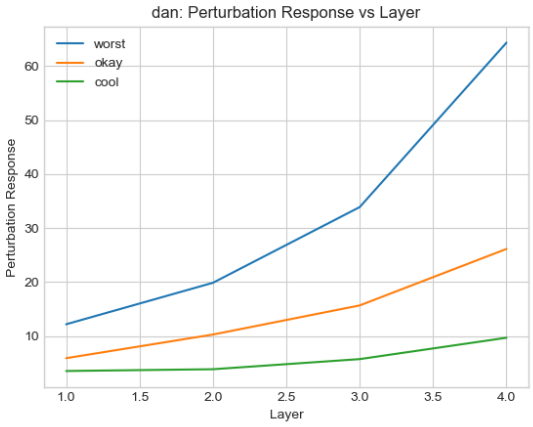
Observations:
The first table displays the accuracy and loss metrics for each layer when applied to the sentiment analysis task. On the validation set, DAN shows slightly better accuracy but that may likely be due to overfitting, but the loss is always more as compared to GRU. When the embeddings are actually applied to the prediction task, both GRU and DAN show similar levels of accuracy (∼86%) with the accuracy for GRU still increasing while the same is decreasing for the DAN model after the 2nd layer.

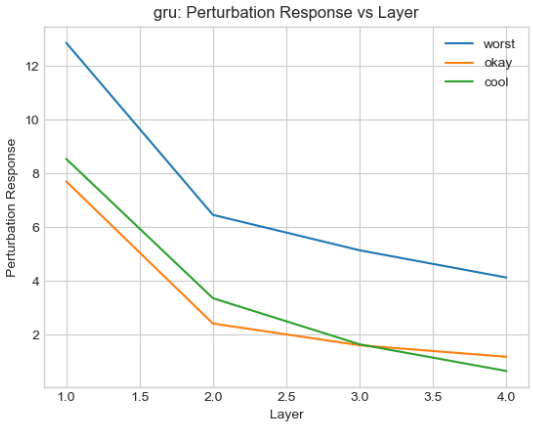
3.2 Probing Sentence Representations for Bi-gram Order Task



Observations:
For the Bi-gram task the accuracy and loss both stay the same (∼50% and 0.69) at every epoch of training indicating how ineffective the model is at predicting the correct order of words. Meanwhile, the GRU models shows increasing progress at every model and eventually ends up with a validation accuracy of 62.61% with a loss of 0.6624. As an extension to the example given in the previous section, consider the bi-gram "Mark Hamill" who is a famous actor, the GRU model perfectly recognizes it as a correct sequence, while the prediction by DAN model fails.

3.3 Analysing Perturbation Response of Representations





Observations:
The plots above represents $|h(orig)–h(perturbed)|$ where $h(orig)$ is the vector representation of the sentence "the film performances were awesome" and $h(perturbed)$ are representations where "awesome" has been replaced by "worst", "okay" and "cool" for different instances of training. Clearly, with the addition of more non-linear layers, DANS show better results at maximising small distinctions as more and more layers are added to the network. As can been seen, "cool" is a word closer to "awesome" so it has the least difference while "okay" being a neutral word and "worst" being negative are further apart. For the GRU encoders the differences are not as pronounced as DANs and in fact, the differences become less for the example sentence when more than 2 layers are used to process the text.

References

1. DAN with dropout