Executive report on Chatbot implementation

MSc Intelligent Systems & Robotics, De Montfort University.
Author: Ibrahim Ahmethan, Supervisor: Dr.Aboozar Taherkhani

This assignment details the creation of a chatbot tutorial using Pytorch and the base code developed by Matthew Inkawhich, which employs a seq2seq model (a type of recurrent neural network) as the core architecture. This model processes input data by converting it into a sequence of words and generates a corresponding output sequence of words, allowing the chatbot to quickly respond to user inquiries. The tutorial utilizes the Cornell movie-dialogs corpus dataset for training, evaluation, and implementation. The report also examines the various factors that influence the development of the default model and compares the improved model to the default model. Additionally, the report discusses any challenges and limitations encountered during the study.

Study aim:

The goal of this study is to understand, develop and expand the default model given in Lab 7 by utilizing Pytorch. The model could be expanded by adapting many different approaches, such as using different datasets, applying various methods and algorithms to preprocess the dataset and utilizing different types of neural networks to build and enhance the system properly. In addition to the default seq2seq model with its Gated Recurrent Neural Network (GRU) embedded in the encoder and decoder of the default model, a Long Short Memory Term (LSTM) and Recurrent Neural Network (RNN) are introduced to expand the default GRU model. The results and comparison tables are displayed in the appendix section with a brief conclusion.

Another aspect of this assignment is hyperparameter fine-tuning, the adopted approach is to test different hyperparameter values based on trial and error alongside previous experience dealing with fine-tuning. Many hyperparameters such as Attention Models, Dropout, Teacher Forcing Ratio, Clip, size of Hidden State, Learning Rate and different kinds of optimizers such as Adaptive Moment Estimation (adam) and Stochastic Gradient Descent (SGD) are tuned and tested. In summary, hyperparameters are tuned carefully with different values and parameters. The results of varying hyperparameters are examined and results are displayed in the following appendix section.

after completing the model and fine-tuning the hyperparameters, in order to examine and evaluate the accuracy of the chatbot, for this purpose two different evaluation method is used, first one is the Bilingual Evaluation Understudy Score (Blue) and Cosine Similarity implemented. The results of the Blue score and Cosine Similarity are printed after each conversation with the chatbot, the conversation examples and results are shown in the appendix section.

Conclusion:

- In this project, GRU and LSTM models showed the best performance, LSTM performed better than GRU at some point.
- The RNN performance proved the fact that the simplicity of the RNN architecture could not handle the complexity and the length of some sequences, although the performance of the RNN sometimes was better than LSTM, therefore more complex neural networks with the ability to save or remember important features such as GRU and LSTM are preferred.
- Enhancing the neural network with attention models was essential to increase the model's performance and decrease the loss function during the training phase.
- The iteration number increased from a default of 5000 iterations to 9000 iterations for all models, **the average loss was calculated by summing the loss function outcome results of each single iteration and dividing them by 9000 to obtain the average loss for each model**. The formula used to calculate the average loss is avg loss = $\frac{\sum_{1}^{9000} loss\ function}{\sum iteration}$
- Various hyperparameters have different effects on different neural network models, hence any hyperparameter change could affect a model in either a positive or negative way.
- The default batch size and the hidden size is tunned with different values, however, according to the results of the models, the default values of batch and hidden size are considered the best tuning values.
- Utilizing CUDA or GPU during the training phase speeds up the training processes by 50 times comparing the usual CPU computation speed process.
- Dealing with a massive dataset requires a serious amount of data preprocessing.
- Bule scoring and cosine similarity are utilized to evaluate the trade-off between the ground truth and the actual output of the chatbot. Cosine similarity sometimes for some unknown reason resulted in zero evaluation.

- **References**

- [1] https://machinelearningmastery.com/category/natural-language-processing/

- [2] https://machinelearningmastery.com/configure-encoder-decoder-model-neural-machine-translation/

- [3] https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

- [4] https://machinelearningmastery.com/attention-long-short-term-memory-recurrent-neural-networks/

- [5] https://machinelearningmastery.com/encoder-decoder-recurrent-neural-network-models-neural-machine-translation/

- [6] https://medium.com/analytics-vidhya/understanding-embedding-layer-in-keras-bbe3ff1327ce#:~:text=Embedding%20layer%20is%20one%20of,word%20embeddings%20such%20as%20GloVe.

- [7] https://www.tensorflow.org/guide/keras/masking_and_padding

- [8] https://machinelearningmastery.com/introduction-to-tensors-for-machine-learning/#:~:text=A%20tensor%20is%20a%20generalization,33%2C%20Deep%20Learning%2C%202016.

- [9] https://www.youtube.com/watch?v=sZGuyTLjsco

- [10] https://ai.stackexchange.com/questions/16133/what-exactly-is-a-hidden-state-in-an-lstm-and-rnn#:~:text=The%20hidden%20state%20in%20a%20RNN%20is%20basically%20just%20like,at%20each%20time%20step%20t.

- [11] https://jmyao17.github.io/Machine_Learning/Sequence/RNN-1.html

- [12] https://pytorch.org/tutorials/beginner/chatbot_tutorial.html?highlight=chatbot%20tutorial

- [13] Evaluation of a chatbot.pptx (by Dr. Aboozar Taherkhani learning material)

- [14] https://lilianweng.github.io/lil-log/2020/04/07/the-transformer-family.html

- [15] https://towardsdatascience.com/illustrated-self-attention-2d627e33b20a

- [16] Graph Convolutional Networks.pptx (by Dr. Aboozar Taherkhani learning material)

Appendix

Attention model tuning

- Two attention models are implemented, the General attention model performed approximately similar to the Dot attention model.
- The GRU performance succussed two show superior results over the LSTM and RNN
- The LSTM and RNN loss function and performance are close to each other, however, the RNN performance was better since it has a lesser average loss function compared to the LSTM.

Attention model	Average GRU loss	Average LSTM loss	Average RNN loss
General	2.5654	3.5569	3.5022
Dot	2.5664	3.5486	3.5105
Concat	Non	Non	Non

Optimizer tuning

- Two optimizers are used to optimize the model during the training phase
- According to the results, it is very clear that the Adam optimizer performed much better than the SGD
- The performance of LSTM with the SGD optimizer is worse than expected
- In conclusion, the default adam optimizer is a good choice to start with.

Optimizer type	Average GRU loss	Average LSTM loss	Average RNN loss
Adam optimizer	2.5654	3.5569	3.5022
SGD optimizer	4.9255	7.2384	5.4401

Learning rate tuning

- In this experiment all the hyperparameter values were kept at default values except the learning rate
- Three different learning rates are used to examine the effect of the learning rate on different deep neural networks.
- The default GRU with the default learning rate of 0.0001 performed better than the LSTM and RNN
- LSTM showed successful performance with a very low loss function compared to GRU and RNN in the case of a large learning rate of 0.001
- LSTM achieved the best results with a learning rate of 0.001 and GRU achieved the best results with a learning rate of 0.0001
- The fastest learning rate lr = 0.00001 (the smallest learning rate) failed to minimize the average loss to an acceptable level in all three GRU, LSTM and RNN neural networks
- In conclusion, different learning rates have various effects on different neural networks.

Learning rate tuning	Average GRU loss	Average LSTM loss	Average RNN loss
lr = 0.001	4.5500	2.1903	3.3010
lr = 0.0001	2.5664	3.5486	3.5105
lr = 0.00001	3.7780	4.4512	4.3019

Dropout ratio tuning

- In this experiment all the hyperparameter values were kept at default values except the dropout percentage
- All three neural networks, GRU, LSTM and RNN performed slightly better at a dropout percentage of 0.05 than the default value which is 0.10
- High dropout value is preferable unless we decided to increase the number of hidden layers in the neural network.
- The dropout values should NOT be increased at high levels while the number of hidden layers is fixed, therefore dropout values should be adjusted according to the number of layers in the neural network

Dropout	Average GRU loss	Average LSTM loss	Average RNN loss
0.05	2.4860	3.5058	3.5086
0.10	2.5664	3.5486	3.5105
0.25	2.7093	3.5105	3.5938

Clip tuning

- In this experiment all the hyperparameter values were kept at default values except the clip values
- The default clip value is 50, however, better performance is obtained for a clip value of 25
- GRU and LSTM achieved lower loss function with a clip value of 25, while the RNN at a clip value of 25 did not show any improvement.

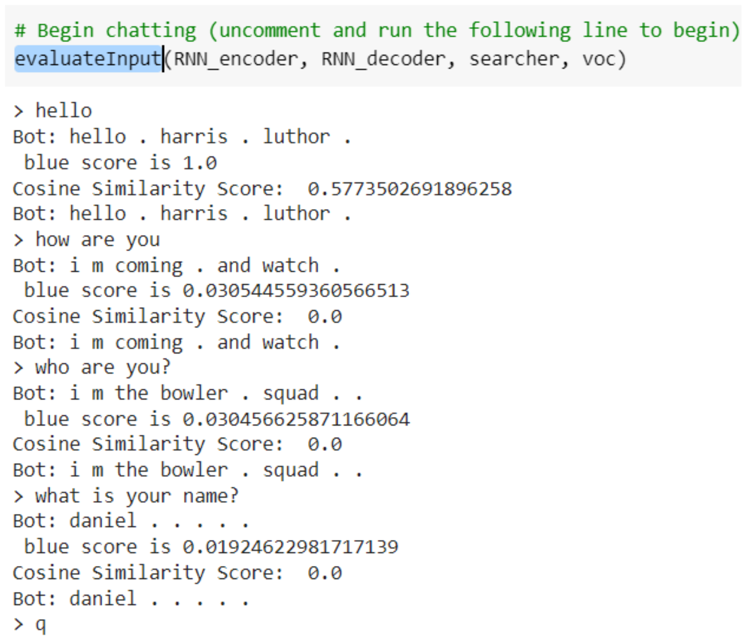
Clip rate	Average GRU loss	Average LSTM loss	Average RNN loss
25	2.5186	3.5112	3.5235
50	2.5664	3.5486	3.5105
75	2.5672	3.5073	3.4540

Teacher forcing ratio

- In this experiment all the hyperparameter values were kept at default values except the teacher forcing ratio.
- Decreasing the teacher-forcing ratio to 0.1 caused all three models to deteriorate, in addition, the loss functions increased by an average of 20% which is considered a very high increase in loss functions
- At the default value of 1 the GRU performed a lot better than LSTM and RNN
- At a value of 10, the GRU performance was superior, the loss function decreased dramatically from 2.5664 to 1.4213
- The performance of LSTM at teacher forcing ratio of 10 was almost the same as teacher forcing ratio of 1, however, the performance of RNN at teacher forcing ratio of 10 is worse than ever.

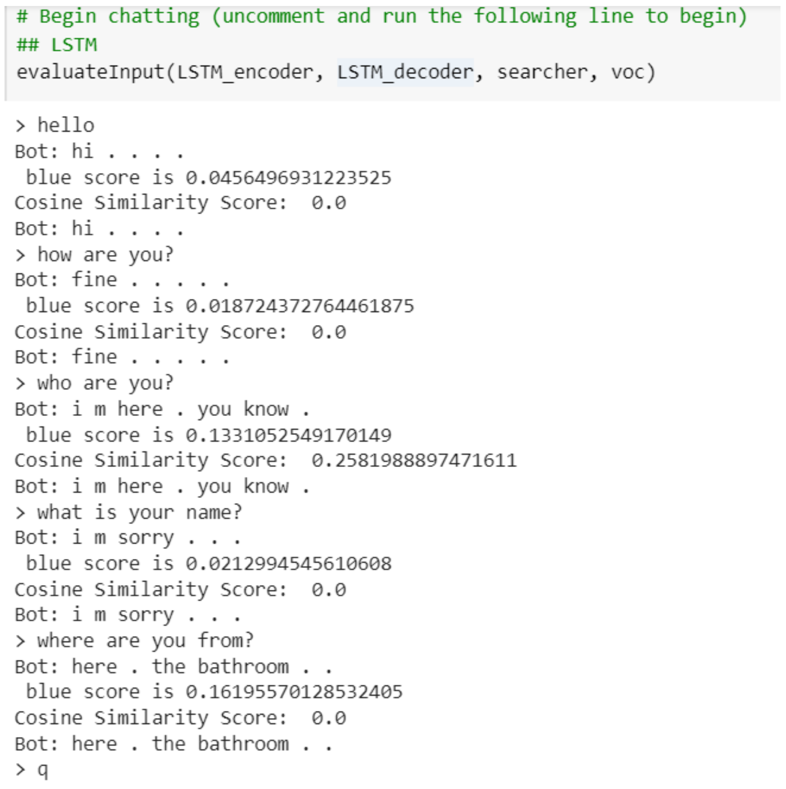
Teacher forcing ratio	Average GRU loss	Average LSTM loss	Average RNN loss
0.1	3.9240	4.4513	4.2966
1	2.5664	3.5486	3.5105
10	1.4213	3.5087	4.4440

- Input evaluation of GRE chatbot model with blue score and cosine similarity results



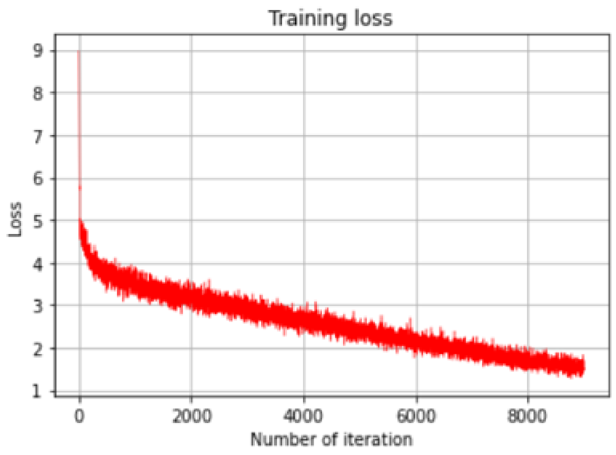
```
# Begin chatting (uncomment and run the following line to begin)
evaluateInput(RNN_encoder, RNN_decoder, searcher, voc)

> hello
Bot: hello . harris . luthor .
 blue score is 1.0
Cosine Similarity Score:  0.5773502691896258
Bot: hello . harris . luthor .
> how are you
Bot: i m coming . and watch .
 blue score is 0.030544559360566513
Cosine Similarity Score:  0.0
Bot: i m coming . and watch .
> who are you?
Bot: i m the bowler . squad . .
 blue score is 0.030456625871166064
Cosine Similarity Score:  0.0
Bot: i m the bowler . squad . .
> what is your name?
Bot: daniel . . . . .
 blue score is 0.01924622981717139
Cosine Similarity Score:  0.0
Bot: daniel . . . . .
> q
```

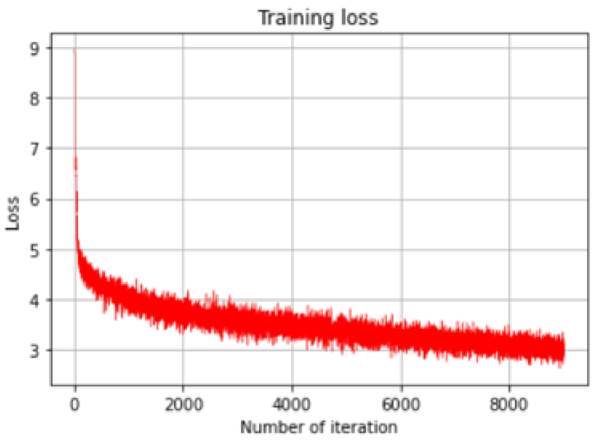
- Input evaluation of LSTM chatbot model with blue score and cosine similarty results

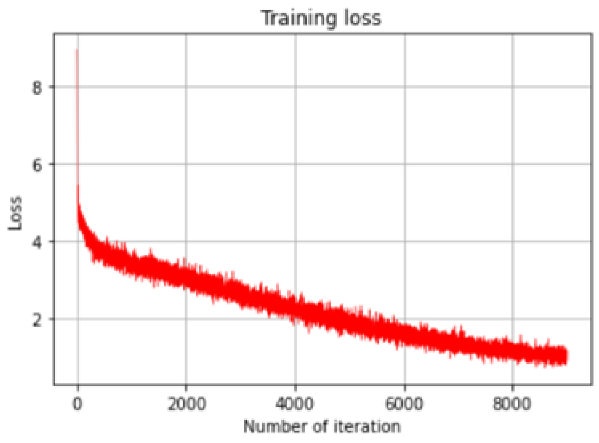


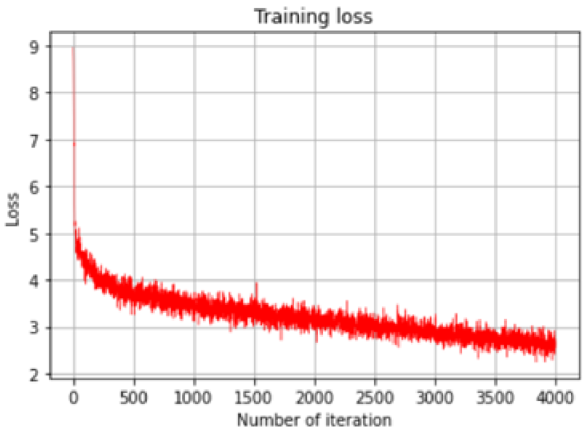
```
# Begin chatting (uncomment and run the following line to begin)
## LSTM
evaluateInput(LSTM_encoder, LSTM_decoder, searcher, voc)

> hello
Bot: hi . . . .
 blue score is 0.0456496931223525
Cosine Similarity Score:  0.0
Bot: hi . . . .
> how are you?
Bot: fine . . . . .
 blue score is 0.018724372764461875
Cosine Similarity Score:  0.0
Bot: fine . . . . .
> who are you?
Bot: i m here . you know .
 blue score is 0.1331052549170149
Cosine Similarity Score:  0.2581988897471611
Bot: i m here . you know .
> what is your name?
Bot: i m sorry . . .
 blue score is 0.0212994545610608
Cosine Similarity Score:  0.0
Bot: i m sorry . . .
> where are you from?
Bot: here . the bathroom . .
 blue score is 0.16195570128532405
Cosine Similarity Score:  0.0
Bot: here . the bathroom . .
> q
```

- Well-performed loss function graphs.
- In most cases, good performance of a model resulted in a stable decrease in the loss function and an average loss function value lower than 3.
- Alongside choosing a robust neural network model, tuning the hyperparameters are essential to obtain the desired results

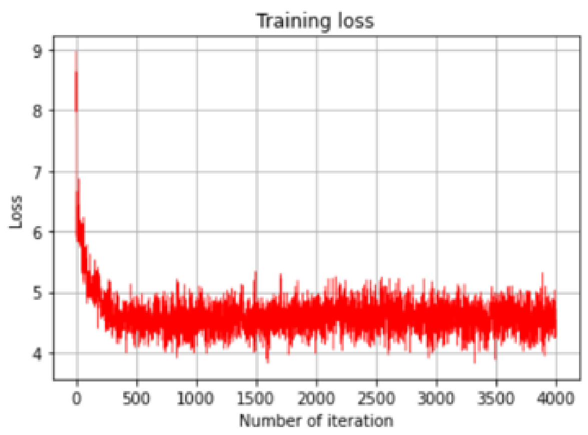


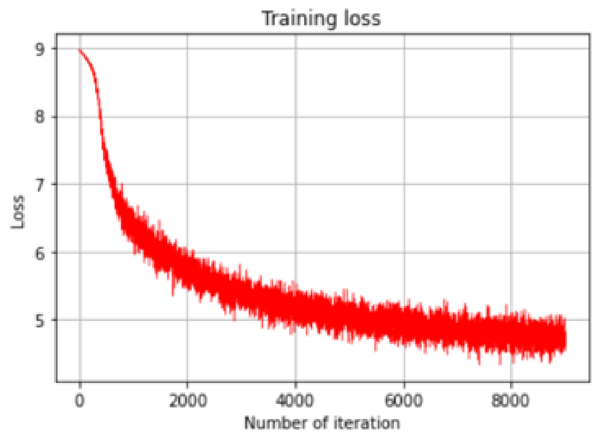


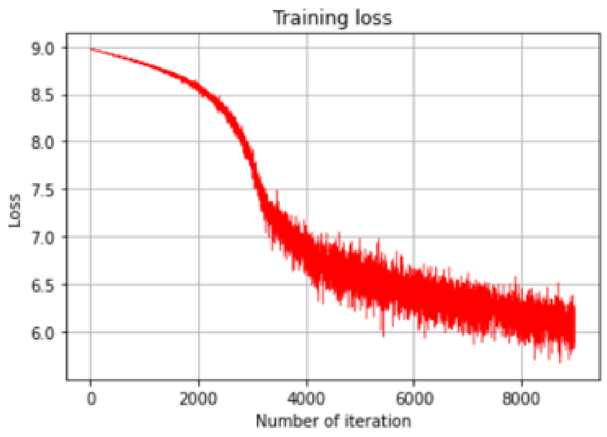


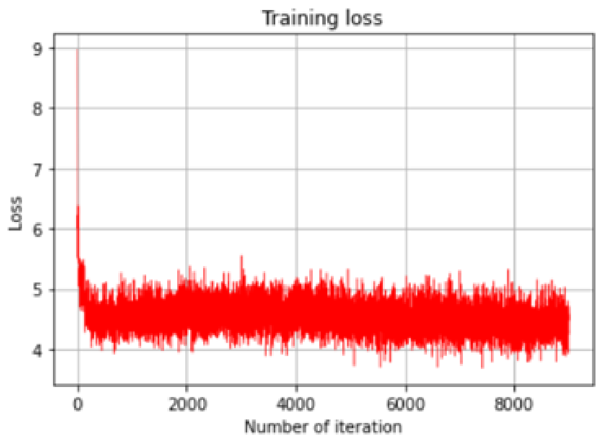


- Bad performed loss functions graph.
- In most cases, the bad performance of a model resulted in a fluctuated and unstable training process.
- A model considered as a bad performance when the average loss function was higher than 3.5
- Fine tuning the hyperparameters is crucial to avoid high loss function.


Training loss
Loss
Number of iteration


Training loss
Loss
Number of iteration


Training loss
Loss
Number of iteration


Training loss
Loss
Number of iteration