Recurrent Sequence-to-sequence Model Development, for Evolving the Veracity of a Conversational AI Platform (Chatbot)

As the motivation of this report, an exhaustive list of adaptations and developments allotted for a conversational artificial intelligence (AI) platform, configured with a generative-based conversational model – *the recurrent sequence-to-sequence chatbot developed by Mathew Inka* [1], is presented as a series of enhancements to the platforms original, provided state. Given the platforms function, to attain intelligent machine-to-human communication using natural language, through the application of Natural Language Processing (NLP), Machine Learning (ML) and cognitive computing (CC) algorithms – *to bridge human and computerized languages for offering human-like interactions autonomously*, the fundamental purpose of this document is to feature said enhancements, and each's contribution to improving the appropriation of the responses compiled and distributed by the chatbot. Through increasing the validity of its responses, encourages intellectual communication to be realised, for any series of input that the chatbot evaluates; this metric can be optimized interchangeably with adjustments issued to the chatbot models' configuration, as the report explores in relation to its evaluation process, which has been utilised for settling the final configuration of the chatbot submitted.

System Refinement

Relative to the recommended but not limited modifications of the chatbot model and for expanding the ways in which the validity of the responses compiled by the model's configuration, could be rectified and evaluated, it was supposed sensible to support multiple corpuses for populating the chatbots vocabulary – *in isolation*, to configure a series of summary tables and graph plots for visually representing model correctness – *accompanied by the models' loss of information and several response accuracy metrics*, and to partition and supplement the dataset populated for the models active text corpus, as training and evaluation data subsets, for realising its capability in pattern generalization; where said capability could then be optimised in use of the graphical outputs mentioned, to identify the effects that the adaptation of other Neural Network (NN) algorithms and adjustments allocated to their hyperparameters, pose on the validity of the responses it produces. Additional to the modifications listed, input error-handling could also be enhanced through integrating a word spell-check and correct algorithm, which justifies the replacement of misspelt and unrecognised words input to the chatbots model, with words known in the chatbots vocabulary to yield responses that are deemed sensical for the intended input that is computed. For which is determined probabilistically by relativizing the semantic similarities of the error, to the words in the vocabulary that is currently populated for the chatbot. Nevertheless, in the proceeding passage features the exhaustive list of the complementary developments issued to the chatbot, where each development is a trialled, functional addition to its original state.

Advocated Development

- **NN algorithm -** Multilayer perceptron (MLP) [2], Long Short-term Memory (LSTM) [3], Elman RNN [4] and Convolutional Neural Network (CNN) [5] (attempted); each of said algorithms have been trialled and purposed for enhancing the recognition of relationships in semantic data, which is applicated for minimizing the error in response generation, for any given input – *relative to the text corpus used to train the model*.
- **Dataset -** Cornell movie-quotes corpus [6] and Microsoft research WikiQA corpus [7]; a series of recognised question-answer focused corpora, that provide a different utterance domain for the chatbot model to be trained and evaluated with.
- **Response calculation -** Determine the response validity for the chatbot model's configuration, using Levenshtein distance ratio [8], sequence matcher similarity scoring [8], cosine similarity scoring [9], Bilingual Evaluation Understudy (BLUE) scoring [10], ordered token-wise similarity scoring and cumulative scoring; where cumulative scoring represents the mean score for all the methods listed, to accommodate the variance in objective that the numerous scoring algorithms implement.
- **Data output -** Graph plot and summary table series, in addition to several console output features for identifying the chatbot models training information loss and series of response validity metrics; where graphical representations utilise data management functionality, used for reading and writing said data to files, located externally – *this enables comparisons between the models configuration and resultant performance to be realised confidently, given that each of the supported text corpuses can only be applicated in isolation, for training and evaluating the chatbot.*
- **Fine-tuning -** NN algorithm hyperparameters, including that of the Encoder-Decoder Recurrent Neural Network (RNN) [11], MLP, LSTM, Elman RNN and CNN (attempted) models; these parameters were adjusted for enhancing the rate at which the chatbot model learns and generates valid output, using the optimisation algorithm: gradient descent [12] – *when training*. Whereas other fine-tuning procedures explored the adjustment of the training-to-evaluation dataset ratio and mode, the number of iterations used for the models training and evaluation processes, the methods for calculating attention model weighting and the active text corpus, for the range of supported corpora (for the entire testing regime, see ***Appendix A***).

Additional Development

- **Dataset proportioning** - 'Ordered' and 'randomised' operations of sentence pair extraction, for populating separate training and evaluation data subsets, that are relative to the original datasets size and configured split ratio, proceeding from the data pre-processing routine; separate datasets are used to effectively estimate the ability of the model's pattern generalization, for data that the model has not been trained with [13].
- **Spell-checking and correction** - Check and correct misspelt or unrecognised words relative to the words known in the vocabulary populated for the chatbot model; correction is purposed to yield responses to input regardless of error, through determining its intended form using Levenshtein's distance algorithm, which generates cycles of permutations for the error – *with an edit distance of two* [14], and appropriates a correction, probabilistically [15].
- **Concatenate apostrophised words** - Concatenate apostrophised words which are split when punctuation characters are stripped during data pre-processing procedures and when populating the vocabulary of the chatbot model. Concatenating words that were formerly apostrophised enables the chatbot to handle said words when input to the model, which enhances the model's response validity by reducing the frequency of word replacements elected by the correction algorithm abovesaid; this is projected in accordance with the algorithms known accuracy limitations, of "80 to 90%" [15].

Development Discoveries

Given the catalogue of developments regulated by the chatbot, it is recognised that the efforts discussed, have been proficient for enhancing the validity of the responses that its model produces. However, it is also acknowledged that the validity yielded by the model could be improved further, given greater volumes of time and computational resources were available. Conclusions drawn:

- The Cornell movie-quotes text corpus is recognised as the dataset that yields the most validity, for all corpora supported.
- The Gated Recurrent Unit (GRU) RNN module when multilayer perceptron NN is not applicated, yields bettered responses.
- The dot-product scoring method of the decoder model's attention mechanism was well-adapted for yielding valid responses.
- The training-to-evaluation data subset ratio '80:20' and sentence pair extraction mode 'ordered', yields more valid responses.
- The activeness of apostrophised word concatenation enhances error-handling for the model and the validity of its responses.

References

[1] PyTorch (2017) *Chatbot tutorial*. [Online] PyTorch. Available from: https://pytorch.org/tutorials/beginner/chatbot_tutorial.html [Accessed: 23/12/20].

[2] HU, J. (2018) *A Simple Starter Guide to Build a Neural Network.* [Weblog] *Towards Data Science*. 20th January. Available from: https://towardsdatascience.com/a-simple-starter-guide-to-build-a-neural-network-3c2cf07b8d7c [Accessed: 23/12/20].

[3] Deep Learning Wizard (2020) *Long Short-Term Memory (LSTM) network with PyTorch*. [Online] Deep Learning Wizard. Available from: https://www.deeplearningwizard.com/deep_learning/practical_pytorch/pytorch_lstm_neuralnetwork/ [Accessed: 23/12/20].

[4] JIA, W. and ZHAO, D. and SHEN, T. and TANG, Y. and ZHAO, Y (2014) Study on Optimized Elman Neural Network Classification Algorithm Based on PLS and CA. *Computational Intelligence and Neuroscience*. [Online] 2014. Available from: https://www.hindawi.com/journals/cin/2014/724317/ [Accessed: 23/12/20].

[5] Github (2017) *spro/pytorch-conv1d-rnn.py*. [Online] Github. Available from: https://gist.github.com/spro/c87cc706625b8a54e604fb1024106556 [Accessed: 25/12/20].

[6] Cornell CIS Computer Science (2020) *Cristian Danescu-Niculescu-Mizil*. [Online] Cornell CIS Computer Science. Available from: https://www.cs.cornell.edu/~cristian/memorability.html [Accessed: 23/12/20].

[7] Microsoft (2020) *Microsoft Research WikiQA Corpus.* [Online] Microsoft. Available from: https://www.microsoft.com/en-us/download/details.aspx?id=52419 [Accessed: 23/12/20].

[8] Stack Overflow (2011) *High performance fuzzy string comparison in Python, use Levenshtein or difflib.* [Online] Stack Overflow. Available from: https://stackoverflow.com/questions/6690739/high-performance-fuzzy-string-comparison-in-python-use-levenshtein-or-difflib [Accessed: 23/12/20].

[9] GeeksforGeeks (2020) *Measure similarity between two sentences using cosine similarity*. [Online] GeeksforGeeks. Available from: https://www.geeksforgeeks.org/python-measure-similarity-between-two-sentences-using-cosine-similarity/ [Accessed: 23/12/20].

[10] BROWNLEE, J. (2017) A Gentle Introduction to Calculating the BLEU Score for Text in Python. [Weblog] *Machine Learning Mastery*. 20th November. Available from: https://machinelearningmastery.com/calculate-bleu-score-for-text-python/ [Accessed: 23/12/20].

[11] ALI, A. and AMIN, Z.M. (2019) Conversational AI Chatbot Based on Encoder-Decoder Architectures with Attention Mechanism. [Online]. Available from: https://www.researchgate.net/publication/338100972_Conversational_AI_Chatbot_Based_on_Encoder-Decoder_Architectures_with_Attention_Mechanism [Accessed: 23/12/20].

[12] BROWNLEE, J. (2016) Gradient Descent for Machine Learning. [Weblog] *Machine Learning Mastery*. 23rd March. Available from: https://machinelearningmastery.com/gradient-descent-for-machine-learning/ [Accessed: 23/12/20].

[13] SHAH, T. (2017) About Train, Validation and Test Sets in Machine Learning. [Weblog] *Towards Data Science*. 6th December. Available from: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7 [Accessed: 23/12/20].

[14] Pypi (2020) *pyspellchecker 0.5.5.* [Online] Pypi. Available from: https://pypi.org/project/pyspellchecker/ [Accessed: 24/12/20].

[15] NORVIG, P. (2016) How to Write a Spelling Corrector. [Weblog] *Norvig*. February. Available from: https://norvig.com/spell-correct.html [Accessed: 23/12/20].

[16] BROWNLEE, J. (2020) *Train-Test Split for Evaluating Machine Learning Algorithms.* [Weblog] *Machine Learning Mastery*. 24th July. Available from: https://machinelearningmastery.com/train-test-split-for-evaluating-machine-learning-algorithms/ [Accessed: 27/12/20].

Appendices

Appendix A:

Model accuracy, adjusting the number of iterations used to train the chatbot model, when using different corpora

- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** does not exceed '20000' given the computational expense and waiting periods for training the chatbot model.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '100', given that each text corpus (when pre-processed) populates varying quantities of sentence pairs for the chatbot model's vocabulary – lowest count is for the Microsoft research WikiQA dataset
- Where the default Encoder-Decoder RNN configuration is used.
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.

Training iteration adjustment, for the corpora supported by the chatbot model				
Corpus	***Cumulative accuracy***			
	T: 100	***T: 1000***	***T: 10000***	***T: 20000***
Cornell movie-dialogs	16.34%	14.46%	14.30%	14.08%
Cornell movie-quotes	20.02%	14.13%	15.25%	15.83%
Microsoft research WikiQA	18.80%	18.43%	20.49%	20.07%

Table 1: Training iteration adjustment testing, for each of the corpuses supported by the chatbot model.

Corpus elected: ***Cornell-movie quotes***

- Although the accuracies compiled suggest that the Microsoft research WikiQA dataset is well-adapted for training the chatbots model, when evaluating the performance of the chatbot model live, the responses generated by the model seemed non-sensical and grammatically deprived; where the chatbot attempts to answer queries, with what appears to be lines from the corpus directly. Alternatively, the Cornell movie-quotes dataset produces responses that are more sensical and expected for a given series of queries, however, in direct correspondence to the accuracies calculated, the dataset supposedly performs poorer than the Microsoft research WikiQA dataset, mostly. Given the potentiality to combine all the corpora supported by the chatbots model, the validity of responses that the model generates, would assume to be significantly bettered, given that more data and separate types of said data, would be available for training the models learning capability. Nevertheless, given the resource consumption restraints posed, multiple corpuses could not be pre-processed simultaneously or accumulatively and therefore, resource availability is a hinderance in the attainment of increased response validity.

Model accuracy, adjusting the active RNN module for the Encoder-Decoder model

- Where the text corpus used to train and evaluate the chatbot model is the Cornell move-quotes dataset (elected from previous test scenario(s)).
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '1000'.
- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** is set to '10000' given the computational expense and waiting periods for training the chatbot model beyond this amount and for the similarity in accuracy to amounts tested beyond.

RNN module adjustment, for the Encoder-Decoder model	
RNN module	***Cumulative accuracy***
	T: 10000
Gated Recurrent Unit (GRU)	16.15%
Long Short-term Memory (LSTM)	16.14%
Jeffrey Elman RNN	15.43%

Table 2: RNN module adjustment testing, for the encoder and decoder models of the chatbot.

RNN module elected: ***GRU***

- Evidently from the results presented above, the GRU RNN module provides the highest degree of validity for its responses generated, when compared to all other RNN modules supported by the chatbot model. When evaluating the performance of the chatbot model live, the responses generated were mostly sensical and anticipated for the series of queries submitted; to note, the responses generated by the GRU RNN module were vastly more valid, when compared to the responses generated in use of the other RNN modules listed, which defers from the response validity calculated for the LSTM RNN module. In which, the LSTM and Elman RNN modules would more frequently influence the chatbot model's response generation, undesirably, where said responses would more often feature duplicate word sequences and punctuation characters, which were regularly recognised as being misplaced also – *thereby their responses were grammatically erroneous*.

Model accuracy, altering the activeness of MLP NN for the encoder models embedding layer, alternating the active RNN module

- Where the text corpus used to train and evaluate the chatbot model is the Cornell move-quotes dataset (elected from previous test scenario(s)).
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '1000'.
- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** is set to '10000' given the computational expense and waiting periods for training the chatbot model beyond this amount and for the similarity in accuracy to amounts tested beyond.
- Where ***S*** is the resultant model accuracy when using the SoftMax activation function, used to convert the word embeddings vector (of real numbers), into a vector of probabilities that are proportional to each's value (relative to their scale in the vector). ***NS*** represents the resultant model accuracy when not applying the SoftMax activation function.

MLP NN activeness, for the encoder models embedding layer, alternating the active RNN module			
RNN module	***Activeness***	***Cumulative accuracy***	
		T: 10000	
Gated Recurrent Unit (GRU)	Active	***NS***: 15.98%	***S***: 13.10%
	Inactive	16.15%	
Long Short-term Memory (LSTM)	Active	***NS***: 15.13%	***S***: 14.91
	Inactive	16.14%	
Jeffrey Elman RNN	Active	***NS***: 16.19%	***S***: 10.29%
	Inactive	15.43%	

Table 3: MLP NN activeness testing, for the encoder models embedding layer when alternating the active RNN module for the encoder and decoder models, of the chatbot.

RNN module and MLP NN activeness elected: ***GRU (MLP NN inactive)***

- In correspondence to the results obtained, the GRU RNN module when the MLP NN is not applied to the embedding layer output of the encoder model, provides the highest degree of validity for its responses generated, when compared to all other RNN modules supported by the chatbot; this outcome considers both states of the MLP NN's activeness. Upon evaluating the live performance of the chatbots model, the responses generated were mostly sensical and anticipated for the series of queries submitted; where said responses generated by the GRU RNN module when the MLP NN is inactive, were more valid when compared to the other RNN modules, irrespective of the MLP NN's activeness, for the encoder models embedding layer output. Meanwhile, upon the MLP NN being applied to the embedding layer output of the encoder model, whilst configured to SoftMax the output tensor of the network, all responses generated by the chatbot model with said configuration were non-sensical and limited to the same sentence, for any given input; thereby, the SoftMax activation function was neglected for the configuration elected.

Model accuracy, alternating the Thang Luong attention mechanism, weight scoring method of the decoder model

- Where the text corpus used to train and evaluate the chatbot model is the Cornell move-quotes dataset (elected from previous test scenario(s)).
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '1000'.
- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** is set to '10000' given the computational expense and waiting periods for training the chatbot model beyond this amount and for the similarity in accuracy to amounts tested beyond.
- Where the active RNN module of the Encoder-Decoder model is 'GRU' (elected from previous test scenario(s)).
- Where the encoder models embedding layer output, is not transformed by MLP NN operations (elected from previous test scenario(s)).
- Where red-filled table fields represent dysfunctional attention mechanism, scoring methods.

Thang Luong attention mechanism, weight scoring method adjustment, for the decoder model	
Weight scoring method	***Cumulative accuracy***
	T: 10000
Dot-product multiplicative combination	16.16%
General weighted combination	14.38%
Concatenated weighted combination	N/A
MLP NN weighted combination	14.59%
CNN weighted combination	N/A

Table 4: Thang Luong attention mechanism, weight scoring method alternation, for the decoder model of the chatbot.

Scoring method elected: ***Dot-product***

- Relative to the findings of the test scenario conducted, it is apparent that the dot-product scoring method was more accustomed to weighting the outputs of the encoder model, for yielding more valid responses than its counterparts; this was also discovered within live interactions with the chatbot, where the responses generated were mostly more reasonable for the series of queries input, when applying dot-product scoring. Said reasoning was advocated by the bettered grammatical correctness of the responses generated and their contextual backing, for the queries input to the model.

Model accuracy, adjusting the hyperparameters of the Encoder-Decoder model

- Where the text corpus used to train and evaluate the chatbot model is the Cornell move-quotes dataset (elected from previous test scenario(s)).
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '1000'.
- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** is set to '5000' given the computational expense and waiting periods for training the chatbot model, for a vast number of adjustment cycles issued per

parameter. In correspondence to the trend previously discovered – *where more training iterations yield higher response validity*, it is assumed that the results which are acquired, also model the response validity for when more iterations are used to train the chatbot model.

- Where the active RNN module of the Encoder-Decoder model is 'GRU' (elected from previous test scenario(s)).
- Where the encoder models embedding layer output, is not transformed by MLP NN operations (elected from previous test scenario(s)).
- Where the 'dot-product' attention mechanism, scoring method, is applied to weigh the outputs of the encoder model (elected from previous test scenario(s)).

Gradient descent clipping adjustment, for the Encoder-Decoder model			
Hyperparameter	***Value***	***Average information loss***	***Cumulative accuracy T: 5000***
Gradient descent clipping	50.0	3.60%	17.34%
	100.00	3.59%	16.89%
	150.00	3.59%	17.62%
	200.00	3.59%	17.39%
	25.0	3.58%	17.30%
	75.0	3.59%	17.14%
	125.0	3.59%	16.64%
	175.0	3.58%	17.14%
	130.0	3.59%	16.72%
	135.0	3.58%	16.60%
	140.0	3.58%	17.96%
	145.0	3.59%	18.34%
	144.0	3.59%	17.00%
	143.0	3.60%	17.29%
	142.0	3.59%	17.86%
	141.0	3.60%	16.91%

Table 5: Gradient descent clipping adjustment, for the encoder and decoder models of the chatbot.

- Where the gradient descent clipping hyperparameter used when training the chatbot model, is set to '145.0' (elected from previous test scenario(s)).

Teacher forcing ratio adjustment, for the Encoder-Decoder model			
Hyperparameter	***Value***	***Average information loss***	***Cumulative accuracy T: 5000***
Teacher forcing ratio	1.0	3.59%	18.34%
	0.95	3.73%	17.32%
	0.9	3.79%	16.95%
	0.85	3.87%	16.84%
	0.8	3.95%	16.69%
	0.75	4.01%	16.86%
	0.7	4.07%	14.12%
	0.65	4.14%	14.11%
	0.6	4.18%	13.65%
	0.55	4.25%	13.68%
	0.5	4.31%	13.73%

Table 6: Teacher forcing ratio adjustment, for the encoder and decoder models of the chatbot.

- Where the gradient descent clipping hyperparameter used when training the chatbot model, is set to '145.0' (elected from previous test scenario(s)).
- Where the teacher forcing ratio hyperparameter used when training the chatbot model, is set to '1.0' (elected from previous test scenario(s)).
- Where red-filled table fields represent no-worded (undecided) responses generated by the chatbot model, for any given input.

Model learning rate adjustment, for the Encoder-Decoder model			
Hyperparameter	***Value***	***Average information loss***	***Cumulative accuracy T: 5000***
	0.0001	3.59%	18.34%
	0.001	5.30%	11.52%
	0.01	N/A	N/A
	0.0005	3.52%	15.76%
	0.0004	3.44%	16.18%
	0.0003	3.34%	15.70%

Encoder-Decoder model learning rate	0.0002	3.39%	16.54%
	0.00001	4.36%	20.02%
	0.00011	3.55%	17.63%
	0.00012	3.52%	16.43%
	0.00013	3.50%	16.94%
	0.00014	3.47%	16.62%
	0.00015	3.46%	15.87%
	0.00009	3.62%	16.87%
	0.00008	3.66%	17.41%
	0.00007	3.70%	17.73%
	0.00006	3.76%	17.57%
	0.00005	3.82%	17.48%

Table 7: Encoder-Decoder model learning rate adjustment, for the encoder and decoder models of the chatbot.

- Where the gradient descent clipping hyperparameter used when training the chatbot model, is set to '145.00' (elected from previous test scenario(s)).
- Where the teacher forcing ratio hyperparameter used when training the chatbot model, is set to '1.0' (elected from previous test scenario(s)).
- Where the Encoder-Decoder model learning rate hyperparameter used when training the chatbot model, is set to '0.0001' (elected from previous test scenario(s)).

Model learning ratio adjustment, for the decoder model			
Hyperparameter	***Value***	***Average information loss***	***Cumulative accuracy T: 5000***
Decoder model learning ratio	5.0	3.58%	16.64%
	10.0	3.39%	15.70%
	15.0	3.39%	14.58%
	1.0	4.1%	17.26%
	2.0	3.87%	17.75%
	3.0	3.74%	17.24%
	4.0	3.65%	17.92%
	6.0	3.52%	16.41%
	7.0	3.48%	16.66%
	8.0	3.45%	16.22%
	9.0	3.41%	16.64%

Table 8: Model learning ratio adjustment, for the decoder model of the chatbot.

- Where the gradient descent clipping hyperparameter used when training the chatbot model, is set to '145.00' (elected from previous test scenario(s)).
- Where the teacher forcing ratio hyperparameter used when training the chatbot model, is set to '1.0' (elected from previous test scenario(s)).
- Where the Encoder-Decoder model learning rate hyperparameter used when training the chatbot model, is set to '0.0001' (elected from previous test scenario(s)).
- Where the decoder model learning ratio hyperparameter used when training the chatbot model, is set to '4.0' (elected from previous test scenario(s)).

Dropout probability adjustment, for the Encoder-Decoder model			
Hyperparameter	***Value***	***Average information loss***	***Cumulative accuracy T: 5000***
Encoder-Decoder model dropout probability	0.1	3.58%	16.45%
	0.2	3.63%	16.91%
	0.3	3.68%	17.24%
	0.4	3.73%	17.28%
	0.5	3.79%	17.16%
	0.01	3.55%	16.51%
	0.02	3.55%	17.31%
	0.03	3.56%	16.52%
	0.04	3.54%	17.17%
	0.05	3.56%	16.95%
	0.06	3.57%	17.07%
	0.07	3.57%	16.85%
	0.08	3.57%	16.92%
	0.09	3.58%	16.91%
	0.11	3.60%	17.04%
	0.12	3.59%	17.30%
	0.13	3.59%	16.84%

	0.14	3.62%	16.85%
	0.15	3.61%	16.47%

Table 9: Dropout probability adjustment, for the encoder and decoder models of the chatbot.

- Where the gradient descent clipping hyperparameter used when training the chatbot model, is set to '145.00' (elected from previous test scenario(s)).
- Where the teacher forcing ratio hyperparameter used when training the chatbot model, is set to '1.0' (elected from previous test scenario(s)).
- Where the Encoder-Decoder model learning rate hyperparameter used when training the chatbot model, is set to '0.0001' (elected from previous test scenario(s)).
- Where the decoder model learning ratio hyperparameter used when training the chatbot model, is set to '4.0' (elected from previous test scenario(s)).
- Where the Encoder-Decoder model dropout probability hyperparameter used when training the chatbot model, is set to '0.12' (elected from previous test scenario(s)).

Hidden state vector size adjustment, for the Encoder-Decoder model			
Hyperparameter	***Value***	***Average information loss***	***Cumulative accuracy T: 5000***
Hidden state vector size	500	3.66%	17.07%
	1000	3.26%	15.85%
	750	3.44%	17.02%
	250	4.00%	18.09%
	550	3.61%	17.54%
	600	3.57%	16.51%
	650	3.53%	17.30%
	700	3.49%	16.44%
	610	3.55%	17.02%
	620	3.55%	16.64%
	630	3.54%	16.34%
	640	3.53%	16.17%
	660	3.52%	17.61%
	670	3.51%	17.03%
	680	3.50%	16.43%
	690	3.49%	16.89%

Table 10: Hidden state vector adjustment, for the encoder and decoder models of the chatbot.

Hyperparameter configuration settled: ***gradient descent clipping –*** *145.0****, teacher forcing ratio –*** *1.0****, Encoder-Decoder model learning rate –*** *0.0001****, decoder learning ratio –*** *4.0****, Encoder-Decoder model dropout probability –*** *0.12****, hidden state vector size –*** *660*

- In correspondence to the results acquired, it is inevitable that the configuration settled for the Encoder-Decoder model hyperparameters, is optimal for the generation of valid responses, more frequently, for any given input when compared to other tested configurations. As for all test scenarios, the resultant configuration also satisfies the more frequent generation of valid responses – *when compared to other tested configurations*, during live interactions with the chatbot; where said responses can mostly be regarded as being sensical and anticipated for the series of queries submitted – *responses were also, mostly, grammatically correct*. However, given the reduced number of iterations used to train the chatbot model, compared to the more optimal amounts discovered in the preliminary test scenarios, the resultant configuration can only be considered optimal for said iterations used and indicative for where more training iterations are concerned. As previously revealed, the application of more optimal quantities was inappropriate for the time and resource constraints posed, for settling a configuration – *within reasonable numerical fidelity*, which would improve the validity of the responses generated and distributed by the chatbot. Moreover, the numerical fidelities applicated for variable adjustment in the test scenarios above, were also nullified by said constraints, which further hinders the settled configurations truth, as being optimal; whereby, through increasing the numerical fidelity of the variables subjected to the testing procedures, the chatbots model would assume additional optimisation and thus, improved validities of the responses that it would yield.

Model accuracy, adjusting the data subset ratio and sentence pair extraction method, for populating separate training and evaluation data subsets

- Where the text corpus used to train and evaluate the chatbot model is the Cornell move-quotes dataset (elected from previous test scenario(s)).
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '1000'.
- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** is set to '10000' given the computational expense and waiting periods for training the chatbot model beyond this amount and for the similarity in accuracy to amounts tested beyond.
- Where the active RNN module of the Encoder-Decoder model is 'GRU' (elected from previous test scenario(s)).

- Where the encoder models embedding layer output, is not transformed by MLP NN operations (elected from previous test scenario(s)).
- Where the 'dot-product' attention mechanism, scoring method, is applied to weigh the outputs of the encoder model (elected from previous test scenario(s)).
- Where the hyperparameter configuration: gradient descent clipping – *145.0*, teacher forcing ratio – *1.0*, Encoder-Decoder model learning rate – *0.0001*, decoder learning ratio – *4.0*, Encoder-Decoder model dropout probability – 0.12, hidden state vector size – *660,* is used for optimising the chatbot model's performance (elected from previous test scenario(s)).
- Where ***E*** is the method of sentence pair extraction, used to populate the separate training and evaluation data subsets, ***S***.

Data subset ratio and sentence pair extraction method adjustment, for populating separate training and evaluation data subsets			
Training-to-evaluation data subset ratio		***Cumulative accuracy***	
		T: 10000	
S: Training	***S: Evaluation***	***E: Ordered***	***E: Randomised***
80	20	17.16%	14.36%
90	10	18.09%	17.47%
70	30	16.07%	15.71%
60	40	15.85%	15.13%
50	50	15.64%	14.89%
40	60	15.56%	16.24%
30	70	13.83%	14.81%
20	80	13.64%	13.76%
10	90	14.13%	14.62%

Table 11: Data subset ratio and sentence pair extraction method adjustment, for populating separate training and evaluation data subsets, for evaluating the chatbot model's response validity truthfully.

Data subset ratio and extraction method elected: ***80:20 (ordered)***

- Based upon the accuracies compiled by adjusting the training-to-evaluation data subset ratio and alternating the sentence pair extraction method, it is apparent that using larger training datasets and smaller evaluation datasets is better for yielding more valid responses – *generated by the chatbot model,* as is the 'ordered' method of sentence pair extraction. However, in practise, when using the highest training data subset ratio tested – *90:10*, it is known for model response validity to be falsely represented, for when vaster amounts of unseen data are handled [16], which can be made relative to the live interactions with the chatbot; where not "enough data in the test set" assumes that the "models performance cannot be evaluated effectively", conversely, there also requires the balance for "enough data in the training dataset for the model to learn an effective mapping of inputs to outputs". Thereby, the ratio settled was '80:20', given its commonality and recognition within existing neural network applications, for generating responses with the next highest recorded validity and for enabling the chatbot model to maintain a more significant training dataset, for possessing a more developed learning capability – *compared to smaller datasets*, which is used to enhance the frequency at which the chatbot model produces and distributes valid responses, for any given input. When interacting with the chatbot live – *for the configuration abovesaid*, the responses generated by its model were mostly sensical, where the responses that were distributed, maintained relevance to the context of the input queries; it is notable that most responses were also grammatically correct, which contributes to their validity even more so.

Model accuracy, alternating the activeness of split apostrophised word concatenation processing, for populating the vocabulary of the Encoder-Decoder model

- Where the text corpus used to train and evaluate the chatbot model is the Cornell move-quotes dataset (elected from previous test scenario(s)).
- Where the 'ordered' mode of sentence pair extraction is used, for populating separate training and evaluation datasets, the ratio used is '80:20'.
- Where the number of iterations used to evaluate the chatbot model, remains constant at '1000'.
- Where ***T*** is the number of iterations used to train the chatbot model, ***T*** is set to '10000' given the computational expense and waiting periods for training the chatbot model beyond this amount and for the similarity in accuracy to amounts tested beyond.
- Where the active RNN module of the Encoder-Decoder model is 'GRU' (elected from previous test scenario(s)).
- Where the encoder models embedding layer output, is not transformed by MLP NN operations (elected from previous test scenario(s)).
- Where the 'dot-product' attention mechanism, scoring method, is applied to weigh the outputs of the encoder model (elected from previous test scenario(s)).
- Where the hyperparameter configuration: gradient descent clipping – *145.0*, teacher forcing ratio – *1.0*, Encoder-Decoder model learning rate – *0.0001*, decoder learning ratio – *4.0*, Encoder-Decoder model dropout probability – 0.12, hidden state vector size – *660,* is used for optimising the chatbot model's performance (elected from previous test scenario(s)).

Split apostrophised word concatenation processing activeness alternation, for populating the vocabulary of the Encoder-Decoder model	
Activeness	***Cumulative accuracy***

	T: 10000
Active	15.27%
Inactive	16.24%

Table 12: Split apostrophised word concatenation processing activeness alternation, for populating the vocabulary used by the encoder and decoder models of the chatbot.

Activeness elected: ***Active***

- Relative to the test scenario results featured above, from a statistical standpoint the inactive state of the apostrophised word concatenation process, for when populating the vocabulary of the chatbot model, assumes the more frequent generation of valid responses for any given input, when compared to the active state. However, when trailed by live interactions with the chatbot, it was discovered that when inputting apostrophised words absent of their apostrophe characters, the spell-checking and correcting algorithm would be invoked in result of said words being unrecognised by the models vocabulary; where the permuted, correctional word representing the intended word input to the model – *determined probabilistically*, would mostly be an unfitting replacement and cause the responses generated by the model, to orientate around external contextual focuses and to be vastly non-sensical for the query submitted. Whereas when the apostrophised word concatenation process was active, a given apostrophised word absent of its apostrophe character, would be processed similarity to any other word known by the chatbot, in which would mostly yield sensical, contextually correct and grammatically correct responses, for any given input.