



Faculty of Computer Science and Information Technology

**Improving Attentive Sequence-to-Sequence Generative-Based Chatbot
Model Using Deep Neural Network Approach**

Wan Solehah Binti Wan Ahmad

**Master of Science
2022**

Improving Attentive Sequence-to-Sequence Generative-Based Chatbot Model Using Deep Neural Network Approach

Wan Solehah Binti Wan Ahmad

A thesis submitted

In fulfillment of the requirements for the degree of Master of Science

(Computer Science)

Faculty of Computer Science and Information Technology

UNIVERSITI MALAYSIA SARAWAK

2022

DECLARATION

I declare that the work in this thesis was carried out in accordance with the regulations of Universiti Malaysia Sarawak. Except where due acknowledgements have been made, the work is that of the author alone. The thesis has not been accepted for any degree and is not concurrently submitted in candidature of any other degree.



.....

Signature

Name: Wan Solehah Binti Wan Ahmad

Matric No.: 19020167

Faculty of Computer Science and Information Technology

Universiti Malaysia Sarawak

Date : 3/8/2022

ACKNOWLEDGEMENT

First and foremost, all praises and greatest thanks to Allah s.w.t for giving me the strength and determination to complete my study while I am going through a difficult phase during this research journey. I also would like to express my gratitude to my supervisor, Associate Professor Ts. Dr. Mohamad Nazim Bin Jambli for giving me all the guidance, advices, and support in my journey to complete this research. The same goes to all people who have indirectly contributed in the way of discussion and knowledge exchanges. Furthermore, I would like to thank Universiti Malaysia Sarawak (UNIMAS) for their financial support under Zamalah Graduate Scholarship.

Last but not least, I would like to dedicate this special thanks to my parents and my family, who have become my backbone, giving me utmost moral support when things get tough and really hard for me to continue this study. I'm grateful to them, especially to my sisters, because of their never-ending support and encouragement, I had a speedy recovery and was finally able to write up my thesis to completion.

ABSTRACT

Deep Neural Network (DNN) is a combination method between two different subfields of Machine Learning application, including the Artificial Neural Network (ANN) and Deep Learning (DL). An example of the DNN model is the Attentive Sequence-to-Sequence (Seq2Seq) model that was first created to tackle a problem setting in language processing. One of the applications is the chatbot model that works explicitly to accurately respond to users' inquiries. Through the years, a chatbot application has seen some improvement, from generating hard-generic responses to more flexible response. The adoption of DNN method into chatbot application produces a new generation chatbot that called as Generative-Based Chatbot. However, it is difficult to create and train a Generative-Based chatbot model that can maintain relevancy of dialogue generation in a long conversation. Hence, this research's objective aimed to propose an optimization strategy based on Structural Modification and Optimizing Training Network for improving the lacking of accuracy of response in the chatbot application, to propose the algorithm enhancement to improve the current attention mechanism in the Attentive Sequence-to-Sequence model and the network's training optimization of its inability to memorize the dialogue history, and lastly, to evaluate the accuracy of response of the proposed solution through data training on loss function and real data testing. The structural modification that is based on a slight modification in Additive Attention mechanism. The method is by adding a scaling factor for the dimension of the decoder hidden state. The other one is the network training's environment optimization that is done through hyperparameter optimization by selecting and fine-tuning high impact parameters which include Optimizer, Learning Rate and Dropout to reduce error rate (loss function). The strategies applied showed that the final accuracy obtained through the training after implementing a modification in the algorithm is at 81% accuracy rate compared to the

basic model that recorded its final accuracy at 79% accuracy rate. Meanwhile, after modification and optimization, the model's performance recorded its final value of accuracy and loss rate at 87% and 0.51, respectively. The result indicates the performance of the optimized model outperforms the baseline model.

Keywords: DNN, Generative-Based Chatbot, Attentive Seq2Seq model, structural modification, hyperparameter optimization

***Penambahbaikan Model Attentive Seq2Seq bagi Generative-based Chatbot
Menggunakan Pendekatan Deep Neural Network***

ABSTRAK

Rangkaian Neural Dalam (DNN) adalah kaedah gabungan antara dua sub-bidang aplikasi Machine Learning yang berbeza, termasuk Artificial Neural Network (ANN), dan Deep Learning (DL). Contohnya, model DNN adalah model Attentive Sequence-to-Sequence (Seq2Seq) yang pertama kali dibuat untuk mengatasi masalah dalam aplikasi pemrosesan bahasa. Salah satu aplikasi tersebut adalah model chatbot yang secara khusus berfungsi untuk melaksanakan tugas memberikan respon terhadap pertanyaan dari pengguna dengan tepat. Selama bertahun-tahun, aplikasi chatbot telah melihat beberapa peningkatan dari segi kelancaran memberi respon daripada menghasilkan respon yang statik kepada respon yang lebih fleksibel. Penerapan kaedah DNN ke dalam aplikasi chatbot menghasilkan chatbot generasi baru yang disebut sebagai Generative-based Chatbot. Walaubagaimanapun, sukar untuk mencipta dan melatih model chatbot Generative-based yang boleh mengekalkan respon yang relevan dalam perbualan yang panjang. Oleh itu, objektif penyelidikan ini bertujuan untuk mencadangkan strategi pengoptimuman berdasarkan Pengubahsuaian Struktur model dan Pengoptimuman Rangkaian Latihan untuk menambah baik kekurangan ketepatan respon aplikasi chatbot, untuk mencadangkan penambahbaikan algoritma model Attentive Sequence-to-Sequence dan latihan rangkaian pengoptimuman ke atas ketidakupayaannya untuk menghafal sejarah dialog, serta akhir sekali, untuk menilai ketepatan respon melalui latihan data yang diukur dengan Loss Function dan ujian data sebenar. Pengubahsuaian struktur adalah berdasarkan pada sedikit pengubahsuaian dalam mekanisme Additive Attention dengan menambahkan faktor penskalaan untuk dimensi jaringan tersembunyi di dalam dekoder, dan pengoptimuman

lingkungan latihan jaringan yang dilakukan melalui pengoptimuman hiperparameter dengan memilih dan menyesuaikan parameter berimpak tinggi merangkumi Pengoptimum, Kadar Pembelajaran dan Penurunan untuk tujuan mengurangi kadar ralat (fungsi pengurangan kadar ralat). Strategi yang diterapkan menunjukkan ketepatan akhir yang diperoleh setelah melaksanakan pengubahsuaian dalam algoritma pada kadar ketepatan 81% berbanding model asas yang mencatat ketepatan terakhirnya pada kadar ketepatan 79%. Sementara itu, prestasi model setelah pengubahsuaian dan pengoptimuman mencatatkan nilai akhir ketepatan dan kadar kerugian masing-masing pada 87% dan 0.51. Hasilnya menunjukkan prestasi model yang dioptimumkan mengatasi model yang sedia ada.

Kata kunci: *DNN, Generative-based Chatbot, model Attentive Seq2Seq, modifikasi struktur, pengoptimuman hiperparameter*

TABLE OF CONTENTS

	Page
DECLARATION	i
ACKNOWLEDGEMENT	ii
ABSTRACT	iii
ABSTRAK	v
TABLE OF CONTENTS	vii
LIST OF TABLES	x
LIST OF FIGURES	xi
LIST OF ABBREVIATIONS	xiii
CHAPTER 1: INTRODUCTION	1
1.1 Research Background	1
1.2 Problem Statement	6
1.3 Motivation	8
1.4 Research Objectives	9
1.5 Research Scope	9
1.6 Research Significance	10
1.7 Thesis Outlines	10
CHAPTER 2: LITERATURE REVIEW	12
2.1 Overview	12

2.2	Deep Neural Network-Based (DNN) Chatbot Model	12
2.3	Existing Work	20
2.3.1	Recurrent Neural Network (RNN)	20
2.3.2	Long Short-Term Memory (LSTM)	22
2.3.3	Sequence-to-Sequence (Seq2Seq) Model	24
2.3.4	Attention Mechanisms	26
2.3.5	Attentive Sequence-to-Sequence Model	31
2.3.6	The Transformer	32
2.4	Chatbot Optimization	34
2.4.1	Structural Modification	34
2.4.2	Hyperparameter Optimization	34
2.4.3	Summary Of The Chatbot Fundamental Theory Appliance In Various Works	35
2.5	Chapter Summary	37
	CHAPTER 3: METHODOLOGY	39
3.1	Overview	39
3.2	Research Flow	39
3.3	Building the proposed DNN Chatbot model	43
3.3.1	Dataset Pre-processing	43
3.3.2	Model's Configuration	45
3.3.3	Implementation of the Proposed Framework	46

3.3.4	Training and Tuning	50
3.4	Summary	54
CHAPTER 4: RESULT AND ANALYSIS		55
4.1	Overview	55
4.2	Training Configuration	55
4.3	Experimental Setup	57
4.4	Implementing the Enhanced Attention (EA)	58
4.5	Optimizing the Capacity of Training's Network	61
4.5.1	Encoder Type Selection	61
4.5.2	Identifying the Best Optimizer	63
4.5.3	Fine Tune Learning Rate	67
4.5.4	Fine Tune Dropout	70
4.6	Baseline Model vs Optimized Model	74
4.7	Evaluation	76
4.8	Discussion	78
4.9	Chapter Summary	80
CHAPTER 5: CONCLUSION		82
5.1	Summary of Research Contributions	82
5.2	Conclusion and Future Works	83
REFERENCES		85

LIST OF TABLES

	Page
Table 1.1 Application of NLP technique	4
Table 2.1 Summary of the Appliance of Fundamental Theory in Chatbot	36
Table 4.1 Proposed enhancement experimental configuration	58
Table 4.2 Accuracy rate Attentive Seq2Seq and EA Seq2Seq	59
Table 4.3 Encoder type experimental configuration	61
Table 4.4 Accuracy rate for encoder type	62
Table 4.5 Default optimizers' configuration	63
Table 4.6 Optimizers experimental configuration	64
Table 4.7 Accuracy rate of optimizer types	64
Table 4.8 Final accuracy rate and loss rate of optimizers	65
Table 4.9 Learning rate experimental configuration	67
Table 4.10 Accuracy rate of learning rate parameter	68
Table 4.11 Dropout experimental configuration	70
Table 4.12 Accuracy rate of dropout parameter	71
Table 4.13 Final chatbot performance experimental configuration	74
Table 4.14 Comparison result of overall model performance	75
Table 4.15 One-Phrase Question Test	77

LIST OF FIGURES

	Page
Figure 1.1 Basic architecture of chatbot model	2
Figure 1.2 Type of response generation model	3
Figure 1.3 Example of AIML tag	4
Figure 1.4 Generative-Based Chatbot	5
Figure 2.1 ANN architecture	13
Figure 2.2 Theoretical Framework of DNN chatbot model	15
Figure 2.3 Analogy of momentum	18
Figure 2.4 Architecture of Basic RNN	21
Figure 2.5 RNN workflow	22
Figure 2.6 LSTM architecture (Olah, 2015)	23
Figure 2.7 Encoder Seq2Seq (Genthial, 2017)	25
Figure 2.8 Decoder Seq2Seq (Genthial, 2017)	26
Figure 2.9 Example of attention representation in a sentence	27
Figure 2.10 Additive Attention overview (Bahdanau et al., 2014)	28
Figure 2.11 An Overview of Scaled Dot-Product Attention in Multi-head Attention (Vaswani et al., 2017)	30
Figure 2.12 Attentive Seq2Seq architecture (Genthial, 2017)	32
Figure 2.13 Transformer architecture (Vaswani et al., 2017)	33
Figure 3.1 Research design	40
Figure 3.2 The flowchart of dataset preparation	44
Figure 3.3 Pseudocode of Modified Algorithm	48
Figure 3.4 Proposed modified architecture	49
Figure 3.5 The loss curve when the model well-learned	51

Figure 3.6	The loss curve when the model over-fit	52
Figure 3.7	The loss curve when the model under-fit	52
Figure 3.8	The loss curve when the model unable to learn	53
Figure 4.1	Accuracy performance Attentive Seq2Seq vs EA Seq2Seq	59
Figure 4.2	Encoder performance Uni-directional vs Bi-directional	62
Figure 4.3	Accuracy performance of optimizer types	65
Figure 4.4	Model performance according to optimizer types	66
Figure 4.5	Accuracy performance of the value learning rate used	69
Figure 4.6	Accuracy performance of dropout value used	72
Figure 4.7	Loss rate according to dropout values used	73
Figure 4.8	Overall model performance	75

LIST OF ABBREVIATIONS

AdaGrad	Adaptive Gradient Algorithm
Adam	Adaptive Moment Estimation
AI	Artificial Intelligence
ANN	Artificial Neural Network
DL	Deep Learning
DNN	Deep Neural Network
EA	Enhanced Attention
FP	False Positive
RNN	Recurrent Neural Network
LR	Learning Rate
NAG	Natural Answer Generation
LSTM	Long Short-Term Memory
NLP	Natural Language Processing
NMT	Neural Machine Translation
RMSProp	Root Mean Square Propagation
Seq2Seq	Sequence-to-Sequence
TP	True Positive

CHAPTER 1

INTRODUCTION

1.1 Research Background

In recent years, especially in the pandemic era, the trend of chatbot application usage has seen incredible growth. According to a study in 2016 by Aspect Software Research, 44% of consumers said they would prefer to interact with a chatbot over a human customer service representative, (Sweezy, 2018), and the numbers are expected to lead its way in coming years.

While it may be a scary coincidence, the chatbot's topic trends increase in these two years due to the Covid-19 pandemic. As if, it becomes essential in the industry as the pressure in maintaining the services while requiring minimal involvement of live interaction between humans is increased significantly in customer services. For example, when the government of Malaysia announced its first movement control order in March 2020, all flights were banned from flying except the essential one. The aviation industry is struggling to handle hectic changes in flight schedules when mass inquiries and complaints come from upset customers.

The situation worsens when the workloads become increased as the workers' capacity at one time is reduced. This example only highlights the situation in the aviation industry, but in reality, every industry suffers from this problem in terms of customer services or management. Every industry needs to develop a solution to automate almost everything by using a third-party app that can be an agent between the parties that gives the services and receives the services so that 'a new norm practices' is convenient for everyone.

Evidently, a chatbot application becomes very much aligned with the solution needed from the industry. But building an intelligent chatbot that can perform efficiently according to user's demands, however, is quite challenging as it requires an algorithm that works effectively in understanding the context of the inputs from the users, text entailment, and language understanding (Wu et al., 2018). So, to achieve a satisfactory functional chatbot, one must understand well the architecture of the chatbot to find the loophole that exists in the current chatbots.

Before going in deeper, it is important to comprehend the definition of chatbot application and how it evolves from traditional chatter bot to deep neural network chatbot. First of all, a chatbot can be classified as an application that builds under the machine learning field, which can be said as the new interest of study in these recent years. It is considered as a Human-Computer Interaction (HCI) model and an artificial intelligence programme (Bansal & Khan, 2018) that simulates a conversation with human users.

The application involves users' intent, the input-output processing, and the response generation method that retrieves the information from the knowledge base. Figure 1.1 shows the most basic architecture of the chatbot model.

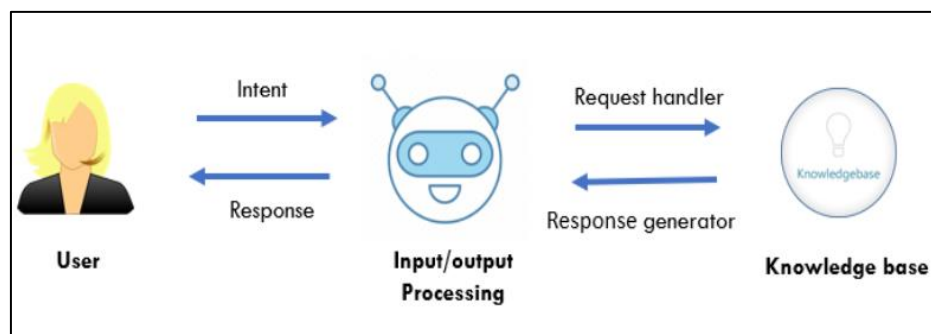


Figure 1.1: Basic architecture of chatbot model

In particular, most of the chatbot development emphasizes the part of the response generation method, and it has two types of methods that indirectly classify the types of chatbot. The first one is categorized as rule-based, and the other one is artificial intelligent-based. Figure 1.2 presents the response generation method's types.

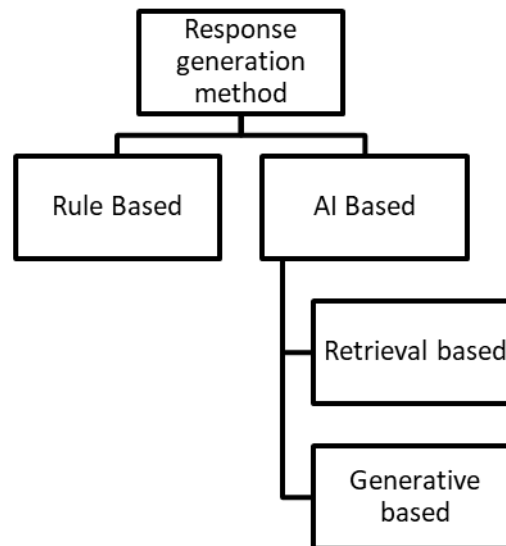


Figure 1.2: Type of response generation model

In the earlier years, a chatbot does not have the capability to generate a human-like response as it lacks language processing knowledge. During those years, the response generation method from the chatbot called a finite state or rule-based, generated hard-generic responses. The method uses a pattern matching technique that is widely derived from Artificial Intelligence Markup Language (AIML) (Trivedi et al., 2019). AIML is considered a knowledge category that consists of input and template that contains answers of chatbot (Ranoliya et al., 2017). Figure 1.3 below shows the example of the AIML tag.

```

<category>
  <pattern> HELLO
</pattern>
  <template>
    Hi, user!!!!
  </template>
</category>

```

Figure 1.3: Example of AIML tag

As the technology evolves with the introduction of Machine Learning (ML) algorithms, the chatbot application has seen an improvement by manipulating language processing. The development of the Natural Language Processing (NLP) method used in chatbots in later years has resulted in a better set of algorithms that successfully changes from a finite state machine response to a retrieval-based response generation method. The illustration of how retrieval-based chatbot works from the application of NLP techniques are shown in Table 1.1 below.

Table 1.1: Application of NLP technique

No	Question	Answer	Entity
1.	What is your name?	My name is Jason.	name
2.	What do you want to eat?	I want to eat fried rice.	eat

The NLP works by matching the predefined entity in the input from users with the predefined answers in the database. A special classification of ‘entity’ makes the chatbot responses more flexible than the finite-state method.

In the recent development of chatbots, most of the current chatbots use Artificial Neural Network (ANN) method combined with a Deep Learning environment as ANN has been proved can generate an action that can imitate a human being. In other words, ANN becomes a ‘brain’ for the chatbot application to learn how to generate a proper response. An example of a popular model that adopts the ANN and Deep Learning method is the Sequence-to-Sequence model. It is widely used in machine translation and chatbot application that involves a language processing.

The existence of ANN and Deep Learning fields then combined with Natural Language Processing (NLP) technique produces a new generation chatbot called a Generative-Based Chatbot. An illustration of the generative based type is present below in Figure 1.4.

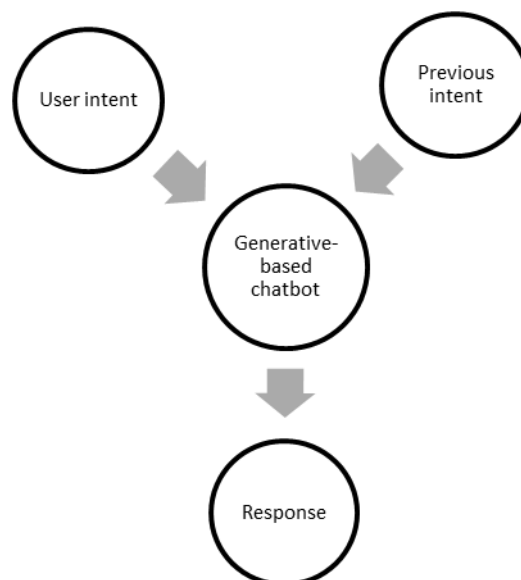


Figure 1.4: Generative-Based Chatbot

A Generative-Based Chatbot is considered as an ultimate high-functional chatbot that can generate a dynamic response without bounding with a hard borderline of domain knowledge by leveraging the Natural Language Generation (NLG) technique (Shang et al., 2015). However, the researchers admit that creating and training such a model is difficult because it requires a large amount of training data to create a successful conversation (Hien et al., 2018). Often in a long conversation, the chatbot start to loss the relevancy in dialogue generation (Sojasingarayar, 2020).

1.2 Problem Statement

As mentioned in the introduction, it is difficult to create and train a Generative-Based chatbot model. Most of the current chatbots applied the existing technology that used different areas and models to produce an almost-like generative chatbot. For example, the hybrid methods that evaluate the retrieved to generated response and select the most effective one (Song et al., 2018). By any means, the current chatbot still has the limitations and can be described as below:

- i. Most existing chatbot models have a bad tendency to generate a generic or inconsistent answer. This is because the current chatbot still heavily rely on the rules that are predefined in the algorithm technique (Nuruzzaman et al., 2018). In general, a chatbot has their own domain knowledge that usually function as a database that store the predefined of question-answer pair. But, even when a huge database is used, there is usually a restriction for the chatbot to recognize the entity needed in their domain knowledge.
- ii. Chatbot model do not perform well on the longer sentences, as the information got lost in the process, resulting in low response accuracy

(Sojasingarayar et al., 2020). In the deep learning field, there is a sub-area that specifically help the application to perceive and adjust the environment it needs to find the focal point accordingly. The sub-area is called an attention mechanism. Attention mechanism becomes a vital tool in chatbot application, as the nature of sentences can have many different meanings according to their focal point. As the current chatbots still rely on predefined rules to identify the entity in the input and the database, the attention mechanism will take part in the process by identifying the context of the sentence in finding the underlying meaning based on the sequence of characters and words in the input sentence. But, there is still a limitation in current model that can only focus on the short distance of word's arrangement in a sentence due to an inability to memorize the dialogue history (Caldarini et al., 2022).

- iii. The capacity of the model's network in training algorithm is not optimum for chatbot training as it still cannot maximize the performance of the model (Elgeldawi et al., 2021). In building a neural network chatbot, a deep learning environment is needed to provide a relationship between the neural response and stimulus that exist in the neural networks. The environment is controlled by hyperparameters in which have the functions to determine the networks' structure and how the networks are trained (Elgeldawi et al., 2021). Every hyperparameter needs to be set before training, and it is important to set the variables with appropriate values. As the environment can vary, how the hyperparameter reacts with the environment will also set a different optimum value. The current chatbot application mostly fails to recognize the optimum capacity of the network environment due to its complex nature.

1.3 Motivation

Several studies have been conducted studying the performance of the response generation method and natural answer generation (NAG) in chatbots. In a recent review done on a topic of Enhancements to the Sequence-to-Sequence-Based Natural Answer Generation Models by Palasundram et al. (2020), it highlights a lot of topic area that is still needed to be improved on various factors. The factors such as insufficient information in questions addressed in another works by Li et al. (2016), Zhou et al. (2018), and Ashgar et al. (2018). The exposure bias's factor addressed by Mou et al. (2016) and Bahdanau et al. (2017). The cross-entropy loss function's factor addressed by Jiang et al. (2019) and Wei & Zhang (2019). Lastly the long sentences' factor emphasized by Shang et al. (2015). The proposed enhancement from various studies that include implementing attention mechanism, Tao et al. (2018), additional encoders, Xing et al. (2017), and beam search, Shao et al. (2017) also stated there. However, despite all of these studies, there is still room for improvement from the current models that are proposed by the researchers. Motivated by the cited works, this study aims to unlock the full potential of chatbot applications by doing further research and advancement in a certain area of neural network chatbot.

Throughout the research, a lot of existing works linked to the DNN chatbot model are reviewed that lead to the conclusion that the DNN chatbot model still can be improved. Hence, the very first idea of this research is to adopt an improvement strategy to the existing model, as in any way, it can open up to a lot of potential of another enhancement method. Inspired from many reviewed works, it is decided that the enhancement in this research can be categorized into two categories, including the structural modification and the optimization of the network training's environment, as both can contribute a significant improvement to the model.

1.4 Research Objectives

The main objective of this research is to improve the chatbot functional capacity by optimizing the capability of the chatbot model in capturing the focus of user intent in the query sentence and optimizing the model network training's capacity. To achieve this, there are specific objectives that need to be done as defined below:

- i. To propose an optimization strategy based on Structural Modification and Optimizing Training Network for improving the lacking of accuracy of response in the chatbot application.
- ii. To propose the algorithm enhancement to improve the current attention mechanism in the Attentive Sequence-to-Sequence model and the network's training optimization because of its inability to memorize the dialogue history.
- iii. To evaluate the accuracy of response of the proposed solution through data training on loss function and real data testing.

1.5 Research Scope

This study focused on a specific neural network chatbot model, which is a Sequence-to-Sequence (Seq2Seq) which was originally proposed for natural machine translation (NMT), (Elbayad, 2020). Chatbot application, also known as a conversational agent, operates via language processing. Thus, the Seq2Seq model would be the best benchmarking model in evaluating future chatbot enhancement. In addition, the scope of enhancement focused on attention mechanisms specifically involving Additive Attention and Scaled-dot product Attention. The attention mechanism is considered as one of the important

components in neural machine translation as of its nature that can help to memorize long source sentences (Galassi et al., 2019). After achieving the enhancement, the model's training is done to test the efficiency of the enhanced chatbot's performance. The training used a dataset from Cornell Movie-Dialog Corpus, which contains a large of metadata extracted from movies and TV-shows scripts. The model's training also involving the Deep Neural Network (DNN) approach, which is the combination of ANN and deep learning.

1.6 Research Significance

The relevance of this study can be described generally as below:

- i. The proposed enhanced attention mechanism implemented in the Seq2Seq chatbot model is based on Additive Attention with slight modification by adding a scaling factor in the hidden decoder component. It is inspired by Scaled-Dot Product Attention that is used in the Transformer model.
- ii. The proposed enhancement improved the capability of the chatbot model in capturing the focus of user intent in the query sentence.
- iii. The selection of the most suitable hyperparameters in the DNN approach will optimized the model network training's capacity.
- iv. The result of the proposed solution analysed based on the model's accuracy that indicates the capability of the chatbot model executing its tasks and through the analysis of loss function.

1.7 Thesis Outlines

This thesis is structured as follows. Chapter 2 discusses an architectural overview of the Deep Neural Network-based chatbot model, where it will detail up the model workflow, the existing works and also the theoretical fundamentals of the deep neural

network approach in identifying out the proposed solution for this research. Chapter 3 describes the methodology of the research following step by step, in which it will present the proposed solution for the chatbot model of enhancement in both algorithm and network's capacity. In Chapter 4, the experimental setup, the results and analysis for each component tested will be presented and compared with the benchmarking of the current existing model. Finally, Chapter 5 will conclude all of the findings and lay out the potential improvement for future research.

CHAPTER 2

LITERATURE REVIEW

2.1 Overview

In this chapter, a detail of literature works based on the selected area of studies involved in this research is discussed. It includes the review of the related work, scaling it into a deeper scope, especially for the intended enhancement, and then explain about the workflow in building a neural network chatbot model inspired by various studies. Then, the next section will present the existing works in sequence, explaining the architecture of the models, the problems, and the solution proposed. There is also a discussion about the chosen criteria for hyperparameter optimization from its complex nature, and lastly, the summary of the findings from overall details in Chapter 2. This overview also serves as the foundation for the following chapter, which uses this summary to construct a theoretical framework.

2.2 Deep Neural Network-Based (DNN) Chatbot Model

Before one can proceed to design the chatbot's neural network model, it is crucial to comprehend what an Artificial Neural Network, or ANN, and Deep Learning are. Hence, this section will introduce the overview of the Deep Neural Network-Based Chatbot model that comprises the main component featuring in the model.

The first component of the model is the application method of Artificial Neural Networks (ANNs). ANN are Machine Learning models that attempt to replicate the operation of the human brain (Grossi et al., 2008), whose structure is made up of a huge number of neurons linked together. To begin with, it is actually a single neuron that makes up the most basic ANN model. The neuron will take the weighted total of its inputs (which

would be the dendrites in a biological neuron), applies a mathematical function to them, and outputs the result.

Individual neurons can be layered on top of one another to produce layers of the desired size, which then be placed next to one another in sequence to make the network deeper. The neurons that do not belong to the input or output layers are considered part of the hidden layers when networks have formed in this manner. Figure 2.1 below depicts a visualisation of how ANN imitating a brain network that represents how the input from users is processed between the layers before generating output to the user.

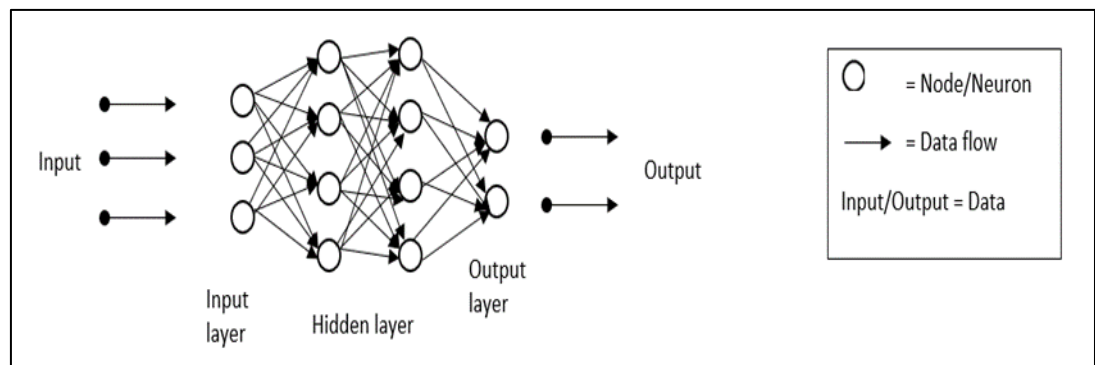


Figure 2.1: ANN architecture

From the figure above, there are three layers in ANN that consist of the input layer, hidden layer, and output layer. Meanwhile, in every layer, there are several nodes or neurons. Each layer uses the output from the previous layers as inputs. In other words, the input data is flow throughout the interconnecting layer by the nodes.

The next component is based on the Deep Learning model that is deeply intertwined with ANN. Basically, Deep Learning is associated with the transformation and extraction of features (Deng et al., 2014) in order to build a link between stimuli and the network's neural

responses. In other words, deep learning actually acts as an environment in assisting the neural networks in executing their tasks. Deep learning implies that it will be using a significant number of layers to extract higher-level features from the data fed to the neural network to improve the performance of the neural models (Vargas et al., 2017).

The two major components work together, namely as Deep Neural Network (DNN), will help the network to progressively learn from the data. In the context of chatbot applications, data processing involves sequential data such as text and audio clips. In this scope, a DNN chatbot model needs a special kind of neural network that is designed to deal with sequential data, called Recurrent Neural Networks (RNN) and its variant is called Long Short-Term Memory (LSTM). The details of these two components will be explained in the next section.

According to the research works from Mallios et al. (2016), Nimavat & Champaneria (2017), Tiha (2018), Nagarajan (2019), there are roughly five general steps in building a chatbot neural network model. From these findings, some similar characteristics are identified and summarized into a framework of a DNN chatbot model that is defined below and is illustrated in Figure 2.2.

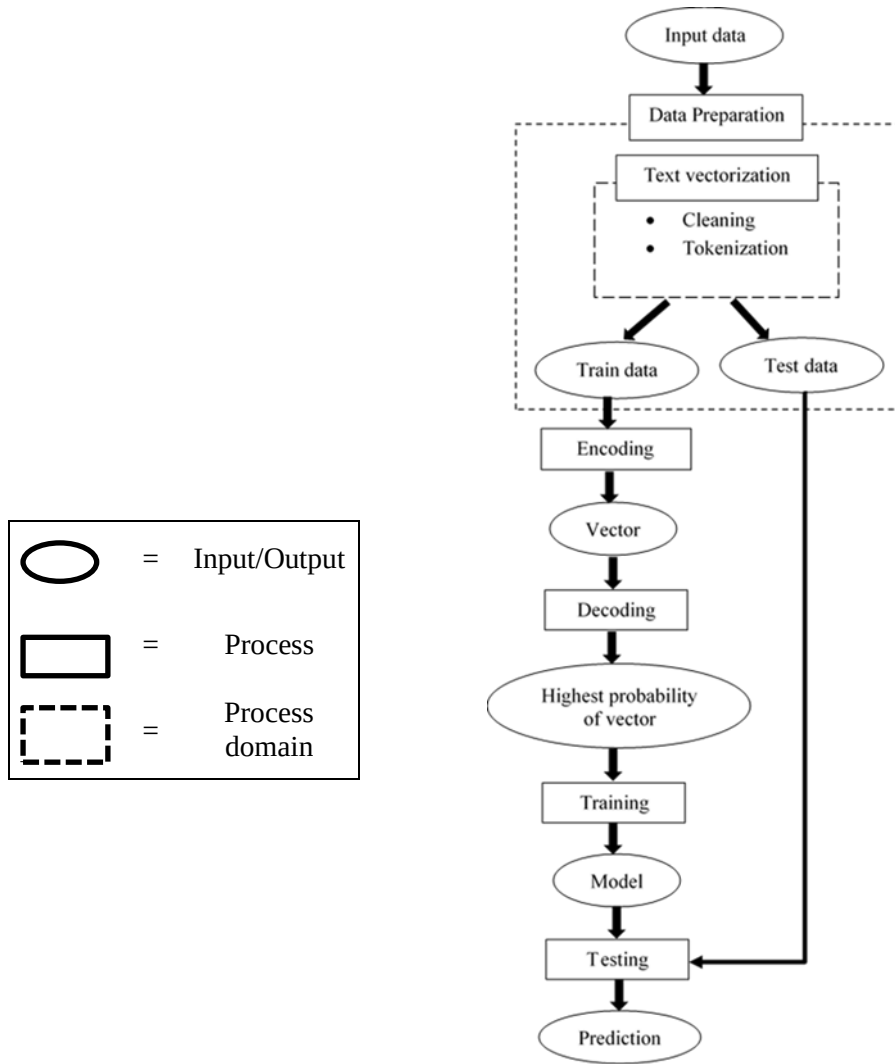


Figure 2.2: Theoretical Framework of DNN chatbot model

The main steps are as follow:

- i. Pre-process and load the data

In the first step of building the model, the data is acquired by selecting the dataset that is fit for the chatbot domain knowledge. After that, the data is prepared by pre-processing and load it into the model. In summary, the data is cleaned and transformed it into a sequence of vector to be used as part of model training.

ii. Define the model.

After preparing the data, an environment of the neural networks will be injected into the model. In this process, the developer needs to specify every neural component in the model's network, such as the number of hidden layers, the input and output size, the encoder and decoder type, and other instances that concludes as part of the algorithm.

iii. Implementation of loss function and optimizer.

The next step in building the model is implementing the loss function and optimizer. The loss function will transform the neural network's environment that is defined in step 2 into a deep learning application. To make it clear, the loss function is a method in evaluating the performance of the algorithm in modelling the dataset by calculating the gradients according to the error prediction in the neural network.

In terms of error prediction, the loss function computes the error for a single training example by measure the discrepancy between the prediction and the desired output (Hajiabadi et al., 2020). Meanwhile, gradients are used to update the weights to fit in the training data. So, it is appropriate to say that the goal of the loss function is simply to measure how well the model is doing on the single training.

An optimizer and the loss function work in tandem to fit the algorithm to the data in the best ways possible. For example, Mean squared error (MSE) is the most basic of the loss function. For each set of weight tried in the models, the MSE is calculated across all input by taking the predictions and

the actual output and then square it before average it out across the whole dataset (Qi et al., 2020). The model then optimizes the MSE functions, make it the lowest possible through the usage of an optimizer.

There are a lot of optimizers that can be used in the neural network training, such as Stochastic Gradient Descent (SGD) (Ketkar, 2017), Mini-batch Gradient Descent (Li et al., 2014), Root Mean Square Propagation (RMSProp), Adaptive Gradient Algorithm (AdaGrad) (Lydia & Francis, 2019), Adaptive Learning Rate Method (AdaDelta) (Zeiler, 2012), and Adaptive Moment Estimation (Adam) (Kingma & Ba, 2015). For the purpose of this research, only three optimizers will be discussed in this research. The selection is on the factor that these optimizers are widely used in the context of deep learning. The optimizers are AdaGrad, RMSProp, and Adam.

AdaGrad uses an adaptive gradient where it has a different learning rate for every single parameter in the model. The strength of AdaGrad is, it will be updating the parameters according to their importance – based on the frequency with which it is updated (Ruder, 2016). The most frequent will be trained with a low learning rate. Meanwhile, infrequently updated parameters are trained with a high learning rate. However, the problem occurred when the most frequent updated parameters could distort the model. The disadvantage of AdaGrad is the learning rate become low after few batches.

RMSProp was created to solve the AdaGrad problem – by exponentially decaying the learning rate. By using past learning rate, RMSProp uses momentum for updating (Mukkamala & Hein, 2017). The analogy of momentum can be described by using the movement of ball rolling

(state of the neural network) down a hill (the graph of loss function), whereas when the longer the ball moves in a certain direction, the faster it is as the ball has high momentum. The momentum is useful to make sure it can move the network pass over local minima and not getting stuck in it, as illustrated in Figure 2.3.

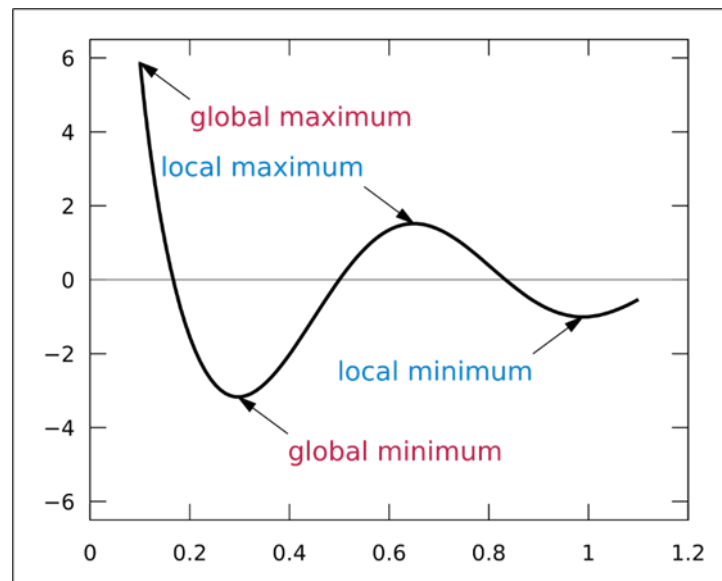


Figure 2.3: Analogy of momentum

Adam works almost the same as RMSProp, the method of using past learning rate to update, except it also use past gradients to speed up learning. Adam basically moves in the network with a great force to make sure it is not suddenly stopped (Kingma & Ba, 2015).

iv. Model training and tuning.

By going into the model training phase, it assumes that all the attributes in a neural network, including all hyperparameters needed and the algorithm is ready to be trained. The term ‘ready’ brings the meaning that all

of the configurations are already complement each other, such as the vector-matrix in the data preparation phase is in the same dimension as the context vector-matrix in the encoder part.

As for the tuning part, usually, it involves experimenting with a set of parameter's values on the model. Usually, the fine-tuning process will be made on a certain hyperparameter that could yield the most significant changes in the result's accuracy.

v. Model evaluation.

Lastly, as in the theoretical summary of related research works, the final phase is on the model evaluation. After a great deal of defining algorithm with specific model's configurations and making some adjustment to the network, the model is trained, and the results are obtained.

From the obtained result, the model's performance analysed according to the values of the model's accuracy, the loss function, and the precision values. If it yields a good performance, then the model will be tested on the test data. There is the prediction for every response expected from the model, and the actual response then will be recorded against the predicted response for the purpose to see if the model capability in answering test questions is on par with the training model's accuracy.

The precision rate for the actual response can be calculated by using the Confusion Matrix. A confusion matrix is a method to measure the performance of a classification model's performance on a set of test data for

which the predicted answer is present. The formula to calculate the precision is as below:

$$\text{Precision} = \text{True Positive (TP)} / (\text{TP} + \text{False Positive (FP)}) \quad \text{Equation 2.1}$$

where,

TP : The actual response is the same as a predicted response.

FP : The actual response is wrong but still in the context.

An extensive description of some of the illustrated steps will be explained comprehensively in Chapter 3 (Subsection 3.3).

2.3 Existing Work

In this section, there are several models will be reviewed, which are Recurrent Neural Network (RNN), Sequence-to-Sequence (seq2seq) model, Attentive Seq2Seq model, and the Transformer model. All of these models are described in order of when it introduced, hence they will show the problems that exist within and how the next model overcomes the problems.

2.3.1 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is introduced to perform a sequence task like natural language in a chatbot applications (Sherstinsky, 2020). It works by having the output from the previous step are fed as input to the current step. To be specific, the basic feature of RNN is that it has at least one feedback connection, allowing activations to loop back. Therefore, this allows the networks to conduct temporal processing and learn sequences.

The architecture of vanilla RNN is shown below in Figure 2.4. In sequential tasks, the current input data is often dependent on the previously applied inputs. Thus, RNN comes

into existence that is tasked with determining the relationship between the present input and previously inputs with the help of a Hidden Layer. The feature within the hidden layer is a hidden state that has the purpose of remembering some information about a sequence.

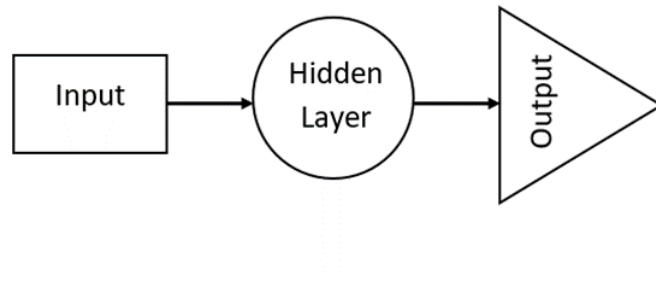


Figure 2.4: Architecture of Basic RNN

In the theory of basic RNN workflow, RNN provides the same weights to all of the activation layers (A) that is associated with hidden layers ($h_{0,1,2,t}$) that has been a break down in Figure 2.5. It will reduce the complexity of memorizing the outputs to be given to the next layer as input by converting the independent activations into dependent activations. Therefore, RNNs can use at any length of information sequence, giving equal importance in processing the sentence as in chatbot application. However, as good as it can be, RNN still suffers a long-term dependency problem and vanishing gradient (Pascanu et al., 2013).

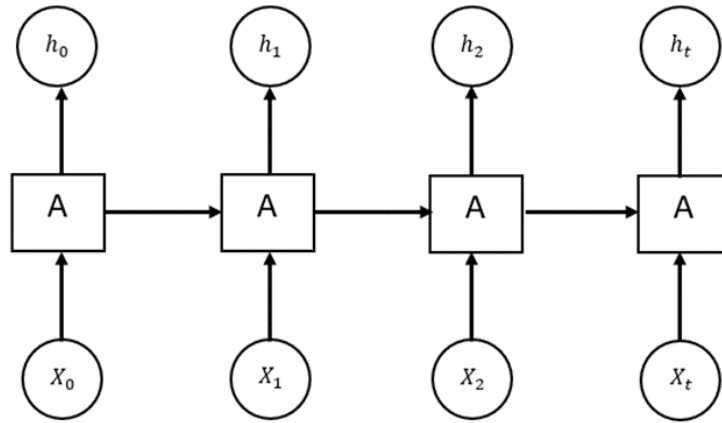


Figure 2.5: RNN workflow

A vanishing gradient can be said as one of the largest barriers in the neural networks, limiting the capability of RNN. Above all, the vanishing gradient is due to the gradient descent algorithm in deep learning that involves calculating the gradient through the learning rate parameter in an iterative optimization algorithm (backpropagation). Fundamentally, the vanishing gradient occurs when the backpropagation algorithm moves back through all of the neurons of the neural net to update their weight (Hochreiter, 1998). This process will go through in a flow shown in Figure 2.5. As it moves back, the gradient calculated deep in the network is ‘attenuated’, causing the gradient to vanish.

To solve this problem, a specific type of neural network is introduced, called Long Short-Term Memory (LSTM).

2.3.2 Long Short-Term Memory (LSTM)

LSTM is introduced in RNN, explicitly designed to remember information for long periods of time. LSTM is unique from vanilla RNNs in a part of the recurring module. It has

a different structure, as can be seen in Figure 2.6. In a single activation layer, there are an additional 4 layers connecting with each other (LSTM cell). The purpose of LSTM is to maintain a cell state that carries information through regulated structures called gates (Hu et al., 2018). Gates has a sigmoid layer that also carries out multiplication operations.

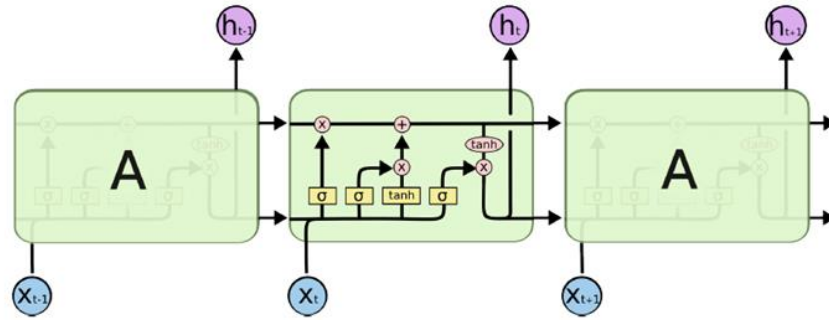


Figure 2.6: LSTM architecture (Olah, 2015)

The first step in LSTM is to let the information through the “forget gate layers”. Then, LSTM decides what kind of information needs to be stored in the cell states by deciding which values going to be updated through the sigmoid layer. Subsequently, a vector of candidate values that are created by the tanh layer would be added to the cell state. These values are defined as the new cell state. Next is the step to decide which parts of the cell state will be the output by running the sigmoid layer. The cell state later will go through the tanh layer, and the output of the sigmoid gate is multiplied. Thus, it is going to be only the important part stored in the output.

From the development of LSTM, it is becoming an inspiration to the born of the Sequence-to-Sequence model.

2.3.3 Sequence-to-Sequence (Seq2Seq) Model

The Seq2Seq model is a great discovery in the field of language modelling. In the context of chatbot application, the Seq2Seq model greatly helps the application thriving its potential on the new marks. Broadly speaking, a Seq2Seq model actually is a problem setting where the input is a sequence and output is also a sequence. In short, a Seq2Seq model helps to transform an input sentence (in sequence form) to a new sequence (target), and both of them can be of arbitrary lengths (Agnihotri, 2019).

A typical Seq2Seq model normally has two parts, including an encoder and a decoder architecture. Both the encoder and decoder are practically two neural networks combined into RNN that using LSTMs modules inside them.

The encoder, as in Figure 2.7, processes the input sentence (train data in the form of sequential vectors), represented as (w_0, w_1, w_2) and compressed into a fixed length of context vector that is also known as sentence embedding. The context vector represents a summary of the meaning of the whole sentence. In the process, the LSTM run over the cell of w_0 before converting it into a hidden state (e_0, e_1, e_2) . The hidden states are then compressed into the encoder representation (e) .

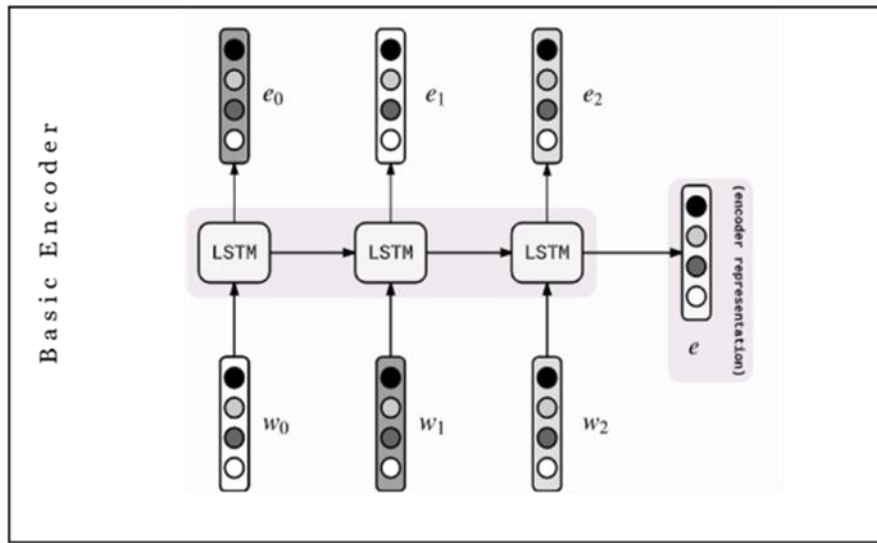


Figure 2.7: Encoder Seq2Seq (Genthiel, 2017)

Meanwhile, the decoder, as in Figure 2.8, works with the encoder representation forwarded from the encoder by making it the initial state in the decoder to emit the transformed output at each timestep (Yu, 2018). The input that becomes a start of sentence vector is represented as w_{sos} together with encoder representation, run by LSTM to computes the next hidden state until it reaches the end of sentence vector, e_{sos} . The process involves the Softmax function that turns a vector value into a probability that remains between 0 and 1. The highest probability value usually is chosen to decide the selection of output words.

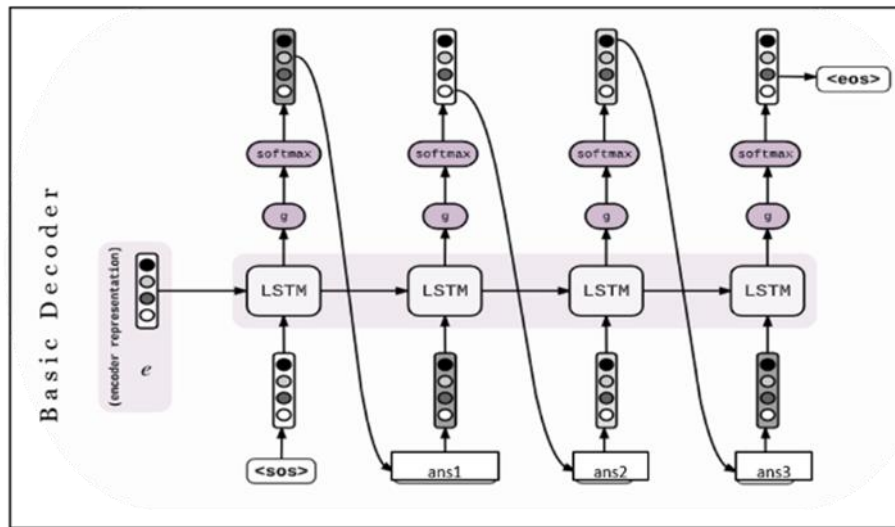


Figure 2.8: Decoder Seq2Seq (Genthiel, 2017)

This encoder-decoder architecture helps to solve the gradient vanishing in RNN architecture by extending the ‘memory’ in remembering the previous inputs. Nevertheless, a critical disadvantage of the Seq2Seq model lies in the fixed-length context vector feature. Apparently, the design cannot deal with the large sequence of sentences because it has often forgotten the first part once it completes processing the whole input. Hence, the attention mechanism was introduced to solve this deficiency.

2.3.4 Attention Mechanisms

In a broad spectrum, attention mechanisms can be defined as a vector that has the weights of importance (attention). The weights of importance are used to predict the word or element in a sentence or in an image. In a sentence according to the example in Figure 2.9, the word “wearing” is co-related with the word “shirt”. It emphasizes that the weight of importance is much higher than the words between “wearing” and “red”.

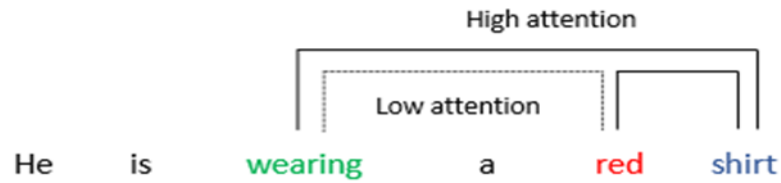


Figure 2.9: Example of attention representation in a sentence

To explain this, in one-sentence, assumed it has a complete structure of the sentence, the algorithm has already identified which one is defined as an entity that can be match up in the database. The word “wearing” is considered a verb entity, so in reality, we expect to see any type of clothes’ word after it, but in language processing, when it meets the type of colour first, it can be confusing. That is why the attention mechanism calculated the weights of importance vector to decide which one of the relationships between words has high attention or low attention value.

There is various type of attention mechanism that differ on how the score of Attention Vector is evaluated, but the pioneer model that first introduced within seq2seq model is Additive Attention Mechanism, proposed by Dzmitry Bahdanau in 2014 in his paper. The paper explains the process of implementing the attention mechanism to improve the seq2seq model in machine translation. The overall process is illustrated in Figure 2.10 below.

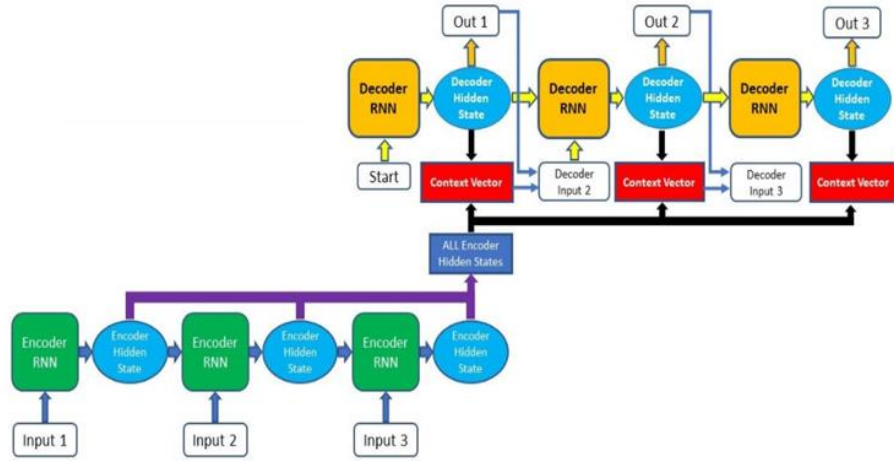


Figure 2.10: Additive Attention overview (Bahdanau et al., 2014)

In brief, there are several important steps in implementing Attention, but the core process lies on the part calculating the Context Vector. In a Seq2Seq model, when the alignment scores are obtained between the previous decoder hidden state and encoder hidden states of each input sequence, the scores are then combined and converted into a single vector and then executed in softmax function. Softmax function works by converting a vector number into vector of probabilities. The formula of calculating the context vector is executed as:

$$\text{score}(\mathbf{s}_t, \mathbf{h}_i) = \mathbf{v}_a^T \tanh(\mathbf{W}_a[\mathbf{s}_t; \mathbf{h}_i]) \quad \text{Equation 2.2}$$

where,

- $\mathbf{s}_t$ = decoder hidden state
- $\mathbf{h}_i$ = hidden state
- $\mathbf{v}_a$ and $\mathbf{W}_a$ = alignment weight matrices
- T = position of output word
- tanh = non-linear activation function

In this process, the alignment scores and the encoder hidden state is multiplied to produce the context vector (Bahdanau et al., 2014). Then to decode the output, the previous decoder output is concatenated with the context vector. Meanwhile, to produce the new output, the output later is fed into the Decoder RNN according to the time step together with the previous decoder hidden states. Gradually, until the output is produced at a specified maximum length, the process of earlier steps repeats itself for each time step.

Ultimately, the Additive Attention Mechanism became a foundation in inspiring more of the incoming attention mechanism that was built on top of it. Presently, the state-of-the-art of attention mechanism was proposed by Vaswani et al., in 2017, adding the new mechanism named as Scaled-Dot Product Attention Mechanism into a “Family of Attention Mechanisms”.

In the same manner, the Scaled-Dot Product Attention Mechanism has the same concept as the original mechanism except for the part of how it calculated the score and scaling it. To be specific, Vaswani proposed this mechanism together with the model called the Transformer model. In fact, Vaswani attention mechanism is purely built for the purpose to cater the uniqueness of the Transformer model that is remarkably different from the seq2seq modelling.

Basically, in Transformer, there is a major component which is the unit of multi-head self-attention that has the role in computing the attention multiple times in parallel. It is shown as in Figure 2.11. The Transformer will encode the input as key-value pairs as encoder hidden states. Subsequently, there is a query that is the product of the compression of the previous output and then produces the next output by mapping the query and the pair of key-value.

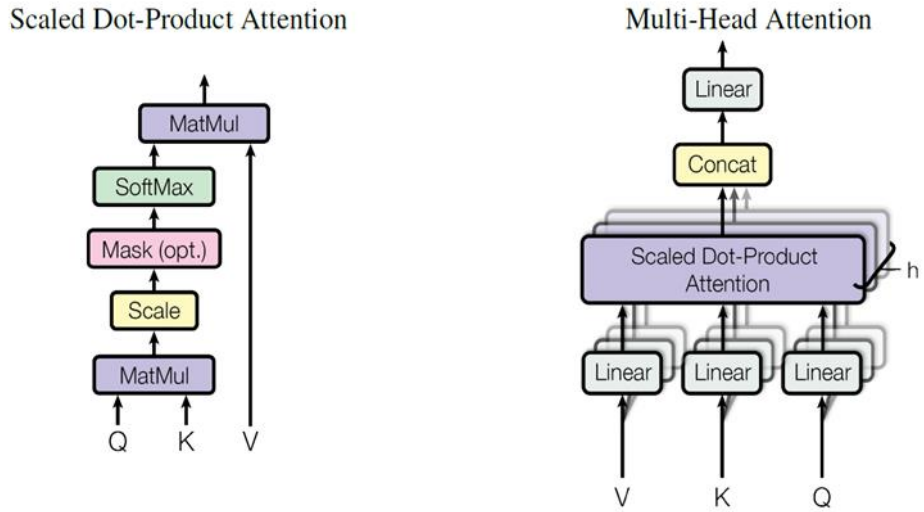


Figure 2.11: An Overview of Scaled Dot-Product Attention in Multi-head Attention (Vaswani et al., 2017)

In summary, the alignment scores are calculated by multiplying the hidden state of the encoder with the hidden state of the decoder and scaled the dimension of the input sequence length (as the source of the hidden state). It is formulated as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{n}}\right)\mathbf{V} \quad \text{Equation 2.3}$$

where,

$\mathbf{Q}$ = compression of previous output as query

$\mathbf{K}, \mathbf{V}$ = encoded representation of the input as key-value pairs

T = position of the output word

$\sqrt{n}$ = dimension of input sequence length

From the explanation in the overview of the Attention mechanism, we can see the technology successfully derived new modelling one after another like has been mentioned above, such as attentive seq2seq modelling and the Transformer. The next section will explain in the summary these modelling that created with attention component as improvement in the history of deep learning modelling.

2.3.5 Attentive Sequence-to-Sequence Model

As have been discussed above, the attention mechanism was first proposed in the vanilla Seq2Seq modelling by Bahdanau et al., (2014) for improvement. By any means, the Attentive Seq2Seq model that is illustrated in Figure 2.12 actually is the same model as the vanilla Seq2Seq model with the additional attention mechanism. Using the concept of Additive Attention, the improvement lies in the implementation of shortcuts between the context vector (C) and the entire source input. The shortcut is the attention weight that calculates the weightage that depends on the timesteps (t) that are most optimal to generate the next word in the sequence (I).

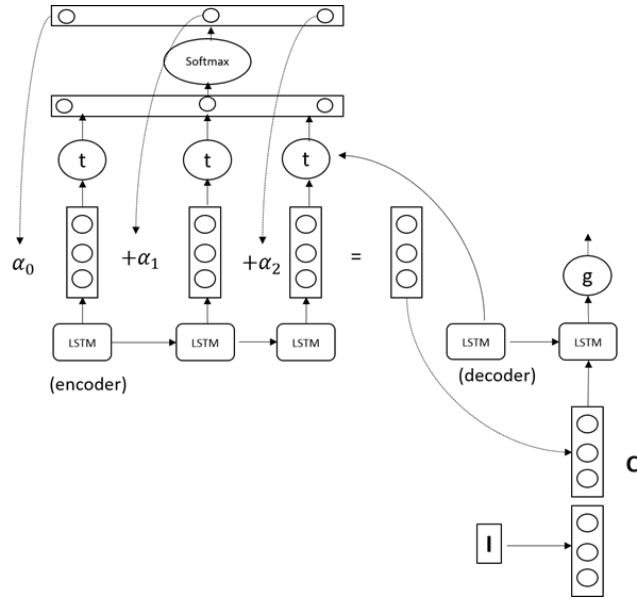


Figure 2.12: Attentive Seq2Seq architecture (Genthiel, 2017)

The context vector is a sum of attention weights and encoder hidden states that map to the input sentence. So, to predict the target word from the source input, the decoder will use the context vector, the decoder output from the previous time step, and the previous decoder hidden state.

2.3.6 The Transformer

Recently, the attention mechanism field has found a great discovery in the paper “Attention is All you Need” (Vaswani, et al., 2017) as it has become the most impactful state-of-the-art model for Neural Machine Translation (NMT) model currently. The paper proposed a Transformer model that purely built on the self-attention mechanism with its own modelling without involving recurrent network units or to put it specifically without using the sequence-aligned recurrent architecture. Figure 2.13 below will show the full view of Transformer architecture with Scaled-dot Product attention as the main component.

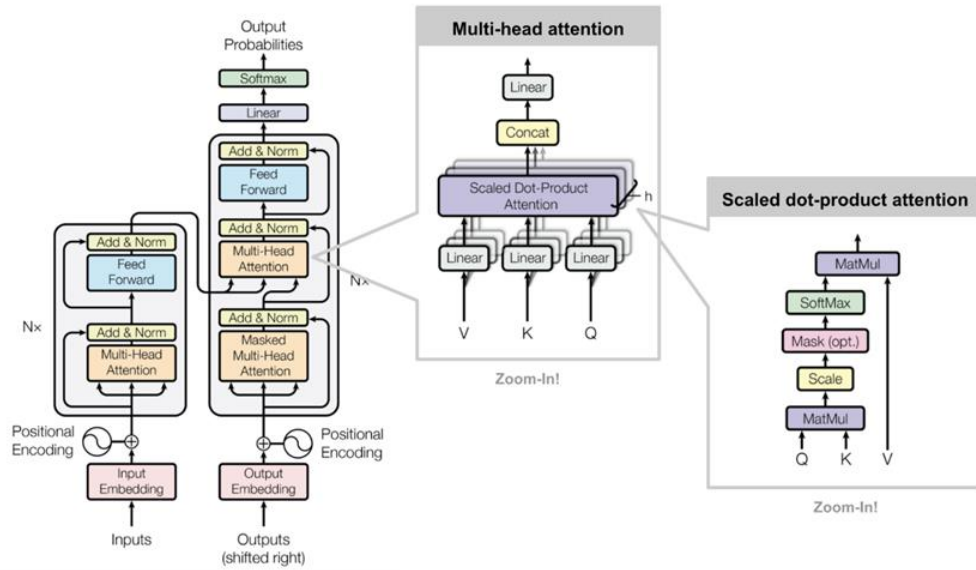


Figure 2.13: Transformer architecture (Vaswani et al., 2017)

Almost the same with Seq2Seq modelling, the Transformer also consists of the encoder and decoder parts. The encoder creates an attention-based representation capable of locating a specific piece of information within an endlessly broad context. It has few identical layers, and each layer presents a multi-head self-attention layer and a simple position-wise fully connected feed-forward network. Meanwhile, the decoder will retrieve information from the encoded representation, and it also has few identical layers. Each layer features two multi-head attention mechanisms and one fully connected feed-forward network sub-layer.

The uniqueness and the efficiency of the Transformer model lie in the component of Multi-head Attention that is connected through residual connection and layer normalization. The multi-head attention consists of the component of Scaled dot-product attention that has been discussed in the section of attention mechanism above.

The Transformer model is proven to perform better than the previous model in Neural Machine Translation (NMT) application. Therefore, Transformer can be applied in the Chatbot model as the architecture for both models are almost the same. However, empirical study according to Hardalov et al., (2018), the Seq2Seq Chatbot model performed better than the Transformer. Hence, this will explain why for the rest of the research, the model that will be used in the Chatbot application as the benchmark model is Seq2Seq modelling.

2.4 Chatbot Optimization

According to the explanation from Subsection 2.2, The chatbot can be separated into two different parts which includes the architecture of the chatbot and the chatbot training. The architecture of the chatbot involves the algorithm used, the flow of the conversation exchanges through natural language processing, and the component to build the chatbot itself. Meanwhile, the chatbot training involving the environment of the network used during the training, usually related to the hyperparameters in the chatbot. Hence, the optimization in chatbot application especially in this research will be focusing on the structural modification on the chatbot's architecture, and hyperparameter optimization.

2.4.1 Structural Modification

Structural modification refers to any modification done to the architecture of the standard chatbot model either in term of adding additional components or modification of the existing algorithm in the model. In this research, the structural modification will be done according to the model that was explained in the Subsection 2.3.

2.4.2 Hyperparameter Optimization

It is crucial to know which hyperparameters to use while training a model. The hyperparameters are referred to as the parameters that are established before the start of

training. The amount of time it takes to train and test a model is determined by the hyperparameters used. Thus, it is important to tune the hyperparameters to make sure the capacity of the neural network model's training environment is optimum (Zheng, 2015). It is because even the slight changes of hyperparameters value can affect the model performance.

In a neural network, the loss function is one of most high impact hyperparameters that has a huge effect on the model's performance. Tuning it in the most optimal environment will resulting in the improvement in the efficiency of how the network executing its tasks (Zheng, 2015). To sum it up, the goal of hyperparameter optimization is to discover the most optimal tuple that minimises a predefined loss function on a given set of data.

Equally, in achieving the main research objective, hyperparameters optimization is considered as one of an important step in improving the chatbot model in terms of the model's training environment. Hence, selecting the high impact parameters would give a significant result to the training.

2.4.3 Summary of the Chatbot Fundamental Theory Appliance In Various Works

The previous sections discussed an overview of chatbot fundamental theory, highlighted the underlying components present in the application that are uniformly applicable in forming a current Deep Neural Network Chatbot. This section would take the next step to summarize the appliance of these theories in actual research mainly focused on the category of improving the chatbot response and hyperparameters optimization.

The purpose is to give an overview of relevant research that has already been done in this field. Table 2.1 presents related works that include the contributing factor to the research, and their proposed enhancement.

Table 2.1: Summary of the Appliarence of Fundamental Theory in Chatbot

	Author, year	Factor	Proposed Enhancement
1	Tao et al., 2018	Attention mechanism in Seq2Seq model	The attention mechanism is divided into: 1. During encoding, the encoder projects the K number of context vectors. 2. During decoding, the decoder scores each of the context vectors and concatenates them using the score as weight and eventually use the final vector for predicting the next word.
2	Shang et al., 2015	Attention mechanism	Adoption of Neural Responding Machine (NRM) into attention mechanism proposed by Bahdanau et al., (2014) by allowing the post representation dynamically change during the generation process.
3	Qiu et al., 2017	Information retrieval and Seq2Seq model	Joint results of Information Retrieval (IR) and Seq2Seq based generation models using Attentive Seq2Seq model.
4	Michael et al., 2019	Attention mechanism and positional encoding	Seq2Seq model with deep CNN-RNN encoder together with RNN decoder
5	Agnihotri., 2019	Hyperparameters	Successive halving by training N/2 configuration number and doubling number of epochs but instead of halting it to do updates, the configurations are run to completion.
6	Hoffer et al., 2018	Neural network training	Algorithm of “Ghost Batch Normalization” allows the generalization gap to decrease although the number of updates not increasing.

2.5 Chapter Summary

In this chapter, various neural network models have been discussed, from the introduction of what kind of DNN Chatbot model is to the literature review of the evolving models and the related works. On the whole, this chapter tries to emphasize the relation between improving a chatbot model with the DNN approach with two major fields, including the structural modification and the optimization of training's environment.

A neural network is a sort of data-driven non-parametric model which became the backbone of the DNN chatbot model. DNN chatbot model mainly built from the appliance of ANN and Deep Learning technology that learns from sequential data. The model that was first introduced in tackling the sequential problem setting is the Seq2Seq model, which is hugely derived from RNN and its variant, LSTM.

Having the Seq2Seq model alone is not capable enough to generate a high functional chatbot in capturing an accurate user intent, so the Attention Mechanism is introduced to improve the model. Attention mechanism was first introduced in Seq2Seq model by using the concept of Additive Attention, then later became a foundation of many attention models out there. The state-of-the-art model is the Transformer model which has its own system without RNN architecture. The component of the attention mechanism inside the Transformer was inspired by the first-generation attention mechanism.

The Transformer model, however, only performs better for Neural Machine Translation (NMT) application and did not outperform the Seq2Seq modelling for a chatbot application. As the niche area of this research involving the chatbot model, the Seq2Seq modelling will be used as the main model. Besides that, this chapter is also highlighting the importance of hyperparameters optimization to cater for the main objective of the research.

Therefore, after selecting the highest impact contributing factor from the related works, it can be concluded that the significant components that will be used for the proposed framework include the Seq2Seq model, attention mechanisms, and loss function hyperparameter. The methodology required to create the proposed framework model will be presented in detail.

CHAPTER 3

METHODOLOGY

3.1 Overview

In the previous chapter, an extensive literature review of the DNN chatbot model, along with its model's architecture, mechanisms, and the training environment, has been discussed. From the discussion, an attentive Seq2Seq model was selected as a benchmark model to be used in the proposed enhancement model as the model capable of learning from the sequential data, besides having the improvement ability on capturing the focus of the user intent. Not only that, the main enhancement would critically lie in the attention mechanism in the decoder part of the Seq2Seq model. In addition, the previous chapter also tries to showcase the equal importance of having an optimal hyperparameter setting in neural network training to improve the model's performance. Equipped with this understanding, a theoretical methodology of the research to create a proposed enhancement model based on the attentive Seq2Seq model will be discussed. In this chapter, the flow in conducting this research is presented first. Each of the elements will be discussed in detail, including the steps in building the proposed model, followed by the presentation of the selection of the components of each contributing factor in the research. Finally, a summary of the chapter will be wrapped up in the last section.

3.2 Research Flow

In building a proposed enhancement model, there are five fundamental phases that include the combination of build and empirical research methodology. The research design is shown in Figure 3.1.

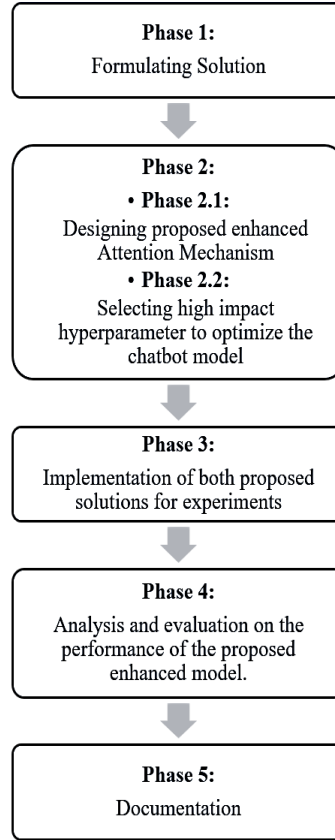


Figure 3.1: Research design

Phase 1 – Formulating solution: In this phase, this study begins with a review of the current model that is widely used in building a DNN chatbot model. As this research niche area is on improving the response accuracy, the process in this phase focusing on finding the factor that can contribute to the significant improvement on the response. Through a series of literature review, various of existing works have been reviewed that lead to the selection of Seq2Seq model as a benchmark model, the Attention Mechanism as part of structural modification, and the hyperparameter optimization as part of training environment to guarantee a significant improvement in the model.

Phase 2.1 – Designing the proposed enhancement in attention mechanism: This phase is concerned with the structural modification in the algorithm selected. Through an

extensive literature review in Chapter 2, it is already justified about the implementation of Additive Attention Mechanism in Seq2Seq model. The review also justifies why the Transformer is not really fit to use in the implementation of the DNN chatbot model. It is because of its architecture that it cannot deal with the dynamic structure of possible intent that comes from users. However, the component of attention mechanism in Transformer model can be used for this research.

On this factor, the designing phase is specifically taking note of the possible combination of the original attention mechanism and the attention mechanism from the Transformer model. Inspired by both mechanisms, a slight modification will be done on top of the Additive Attention mechanism by taking a small contributing component from the Scaled-Dot Attention mechanism to be applied on.

Phase 2.2 – Selecting high impact hyperparameter to optimize the chatbot model: Still in the same stage of proposing an enhancement to the model, that also covered in Chapter 2, the next step is selecting the important parameters for hyperparameter setting to be implemented in the model's training environment. It is already decided from the conclusion in Chapter 2 that loss function stability is one of the training's factors that can offer a significant improvement to the model.

Phase 3 – Implementation of the proposed solutions: After designing an improvement to the mechanism and selecting the impactful parameters in a hyperparameter setting, the implementation of the proposed enhancement will be taking part. This phase is described in detail in Chapter 3, specifically in Section 3.3. A vanilla Seq2Seq model and Attentive Sq2Seq model built first before recording all of the initial results from the training test. The modification design applied to the model. After that, the parameters that were specifically selected implemented that later would going through a fine-tuning process.

Phase 4 – Analysis and evaluation: This phase is covered in Chapter 4 and for each trained model, the loss function results used to evaluate the model performance and accuracy of response yield by the model. The commonly used evaluation metrics in evaluating a chatbot model are the loss rate, the accuracy in part of the loss function, and the precision of the prediction answer in part of the confusion matrix when testing the capability of the chatbot answering the query from the users.

To evaluate the model performance, the common approach involving the comparison of the training's result between the base model, which is considered as a benchmark, and the enhanced model. The comparison's result indicated whether the implementation of the proposed enhancement strategy is successful or not in improving the model's performance. The analysis is done on every experimental that consist of different configuration.

The testing mechanism involves three crucial parts which are the loss function, the confusion matrix and the one-phrase question test. The loss function takes part in measuring the model's training accuracy, while the confusion matrix takes part in evaluating the model's capability in answering test questions.

A good indicator in an obtained result of the loss function is when the loss rate value is decreased over the epoch until it reaches an optimal point through the cross-validation method, and the accuracy value is increased.

For the purpose of further analysis, a one-phrase question test is conducted to validate the training's accuracy result. The result obtained is measured using the confusion matrix method that also known as an error matrix calculation based on the Precision, Recall, and F measure concept. It described the performance model on an arrangement of test information

that already justified the correct value, which is in this case study, referred to the correct prediction answers obtained from one-phrase question test.

Phase 5 – Documentation: All of the designs, experimental findings, and analyses are compiled and documented in the form of dissertations and research papers.

3.3 Building the proposed DNN Chatbot model

Building the proposed DNN chatbot model is a part of methodology for this research. Hence, in this section, the theoretical research flow of Phase 3 is described accordingly in a real DNN chatbot model building process by referencing the steps from a framework that has been briefly summarised in Chapter 2, Section 2.2.

3.3.1 Dataset Pre-processing

To start building a chatbot model, a textual dataset is used, featuring Cornell Movie-Dialog Corpus. This corpus extracted large metadata of fictional conversations from raw movie scripts. It contains 220,579 conversational exchanges, and 304,713 utterances. The data size used in the experiment of this research is limited to 50,000 words considering the capacity that the training machine can support. As the chatbot model heavily uses Natural Language Processing (NLP) technique, the dataset plays a vital role to ensure the capability of the model to understand every question input by the users. Before anything else, preparing dataset into a good high-quality machine-readable data is the first important step that cannot be skipped. The explanation below is based on Figure 3.2.

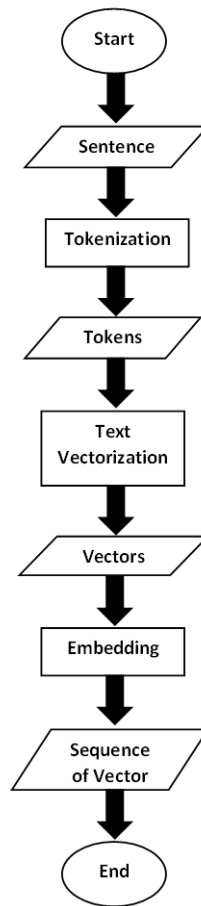


Figure 3.2: The flowchart of dataset preparation

The dataset is prepared by using two basic methods, which are text cleaning and tokenization. These methods fall under data preparation that is a must in the pre-processing textual datasets. Text cleaning is the process of removing unnecessary words and symbols that do not convey any meaning to the model, such as punctuation's, stop words and converting the text into lowercase. In the experiment, the requirement length set for every pair of questions and answers to be trained after doing the data cleaning is limited to 20 words only.

The purpose of text cleaning is to make sure the model be able to retrieve important keywords only. Here is why it is important. Chatbot model totally relies on the trained dataset to recognise specific intent because, in human language, one intent can be represented as in many forms of sentences. In order to understand the questions from the user, the chatbot must have the ability to distinguish a specific intent only. Hence, the text cleaning is to ensure only important words are left for the model to analyse.

Tokenization is a method to convert the text into a set of tokens. Tokenization is basically the division into smaller units of an expression, sentence, paragraph, or whole text documents, such as individual words or terms. For example, the sentence “**How are you?**” is tokenized into [**How**, **are**, **you**]. During the tokenization, the tokens will be mapped into vector positions that later will be embedded as a sequence of vectors. Every vector represents one word.

The dataset then divided into two parts which are categorized as ‘Train Data’ and ‘Test Data’. ‘Train Data’ is used in the whole development, while ‘Test Data’ is used during the testing and evaluation part. From the total of 50,000 words, 70% of data purposely ‘Train Data’, and another 30% is for ‘Test Data’.

3.3.2 Model’s Configuration

First of all, as for this research scope, the Seq2Seq model is built into the network. Hence, the model configurations will include part of Recurrent Neural Network (RNN) in addition to LSTM units that consist of the encoder-decoder architecture. In general, defining the model also included the selection of encoder-decoder type.

The encoder has its own type that allows the developer to choose which one works the best with their model. For encoder type, the developer can choose to work with a unidirectional encoder or bi-directional encoder. The unidirectional encoder is simply a simple RNN that uses LSTM to preserve the information from its last state as inputs.

For the bi-directional encoder, it consists of two independent encoders. One encoder will encode the normal sequence of the sentences, while another one will encode from the reversed sequence. A bi-directional encoder by other means can preserve the information from both directions. The selection of the encoder is based on the result of the experiment that executed in Chapter 4, Section 4.5.1, to compare the performance of the encoder types.

Meanwhile, there is a lot that took place in the decoder part, such as the implementation of attention mechanism and inference mode like beam search. The enhancement of the algorithm is defined here. For the selection of attention mechanism, it is decided to use Additive Attention proposed by Bahdanau et al., (2014) as the mechanism of attention in the network.

Subsequently, the next explanation will be highlighting the main component of the research, implementing the proposed solution for the DNN chatbot model – attention mechanism modification and in addition of high impact hyperparameter selection.

3.3.3 Implementation of the Proposed Framework

In this stage, all of the DNN chatbot model's components implemented as usual, except the additional part, which is a slight modification part that is done in the selected algorithm and fine-tuning the parameters in the selected hyperparameters.

Proposed enhancement in attention mechanism: The main idea of implementing the enhancement attention mechanism is inspired by the mechanism of Scaled-dot product Attention. The modification's method is by taking a contributing component from the mechanism to be added in the Additive Attention algorithm that is already decided beforehand as a main mechanism to be implemented in the network of basic Seq2Seq model.

In this part, a scaling factor that scaled the dimension of the decoder hidden state is added in the decoder transition between the hidden state and the Context Vector. The implication of adding this factor influenced how the score of attention is calculated according to the probability value produced after softmaxed. The algorithm can be described as follows:

Modified Algorithm: Training a Seq2Seq Model with Enhanced Attention Mechanism for Answer Generation.

Input: Question (X) – Answer (Y) pairs, No. of Epoch (E)

Steps:

For epoch 1 to E,

For the batch of question-answer pairs, X and Y, Do

1. Encoder: Encodes every word in the question, then generate the hidden states.

2. Decoder:

2.1 Every token with the highest probability is generated by feeding weighted hidden states from the Encoder.

2.2 The dimension of the hidden state from the decoder is scaled for every token generated from step 2.1.

2.3 Steps 2.1 – 2.2 are executed until the maximum answer length is reached or the token generated is reaching at the end.

2.4 Join all tokens to generate an answer, (Y').

3. The cross-entropy loss (the difference between Y and Y') is calculated.

4. The model parameters are updated.

End For

End For

Output: Trained Enhanced Attentive Seq2Seq model

Figure 3.3: Pseudocode of Modified Algorithm

To give a clear view, Figure 3.4 depicts the proposed enhancement that is added in the basic Additive attention mechanism.

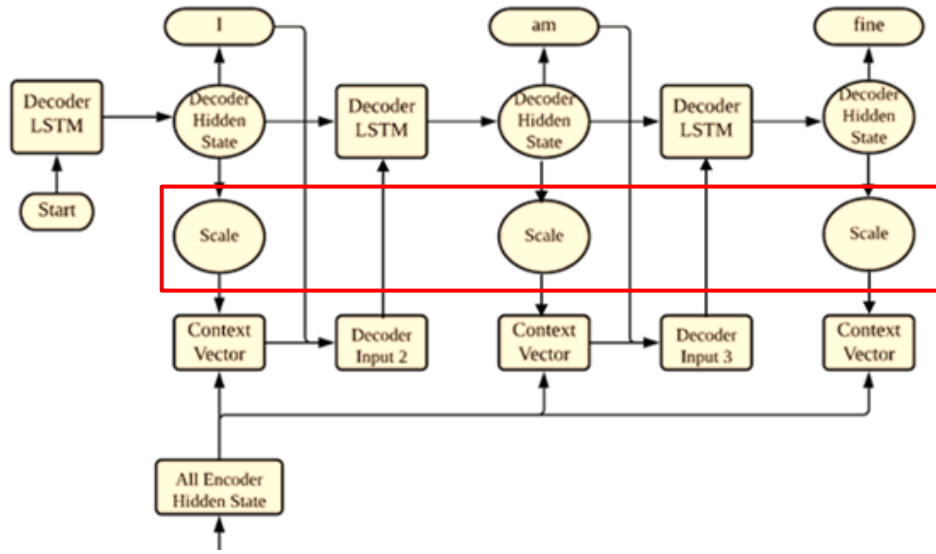


Figure 3.4: Proposed modified architecture

As have been explained, a scaling factor is added, as highlighted in the red box. In a usual Attentive Seq2Seq model as in Figure 2.10, a softmax score is directly computed into the decoder hidden states to be compressed as a context vector. This method is not considering the possibility of having a wide gap of gradients in the output of the probability scores. This factor could lead to the neural network having high confidence prediction when that was not the case. Hence, by adding the scale factor, the stability of gradients can be created.

Hyperparameter selection: Besides that, the implementation of high impact hyperparameter is also included in this phase. As the selection of the hyperparameter involving the loss function, the parameters implemented notably are the Optimizer, Learning Rate, and Dropout.

The optimizer is the driver of training for neural networks. It will help the machine learn about it. That is why optimizer is included in the list of crucial hyperparameters in a

neural network. Optimizer works by applying the gradient to a neural network, and most of it calculate the learning rate automatically.

However, for the sake of optimal optimization, the learning rate also will be tested independently to see if it could optimize the optimizer in which also indirectly could be improving the model accuracy. It is because learning rate can control how quickly or slowly a neural network model learns a problem (Andrew et al., 2013), which is become the important factor in why learning rate is chosen as one of the high impact parameters.

As for the dropout, the dropout is a regularization technique for reducing overfitting in deep learning neural networks by preventing complex co-adaptions of training data (Srivastava et al., 2014). A conventional method to reduce overfitting is assembling a neural network with different model configurations. However, this method is expensive because it needs to train and maintain multiple models. Therefore, training a single model is much more ideal.

A single model can be trained with a large number of different network architectures but can quickly overfit a training dataset. To reduce this, the dropout approach will be used to randomly dropping out nodes during training.

3.3.4 Training and Tuning

This training phase is mainly about setting the number of epochs that are going to fit the model. Basically, one epoch is when an entire dataset is passed forward and backwards through the neural network only once. The parameter in measuring the results depends on the learning rate, meaning it is must be iterative to get the most optimal results. So, the rule

of training the model is the epochs must be within an appropriate range to avoid the case of overfitting or underfitting results.

The model's training nature will always indicate an increased performance. It is always going upward in accuracy rate as high as the value of epoch defined until it reaches its limit. By theoretical hypothesis, the higher value of epoch, the lower loss rate will be yield by the model until the certain point it declines by itself. Hence, the graphs below will be the guidelines to make sure the results are in an optimal state.

Figures 3.5 - 3.8 depict instances of learning curves, with the y-axis indicating the Loss Function and the x-axis showing the number of epochs. The intended selection is a set of parameter settings that permits the model to produce a curve pattern similar to Figure 3.4.

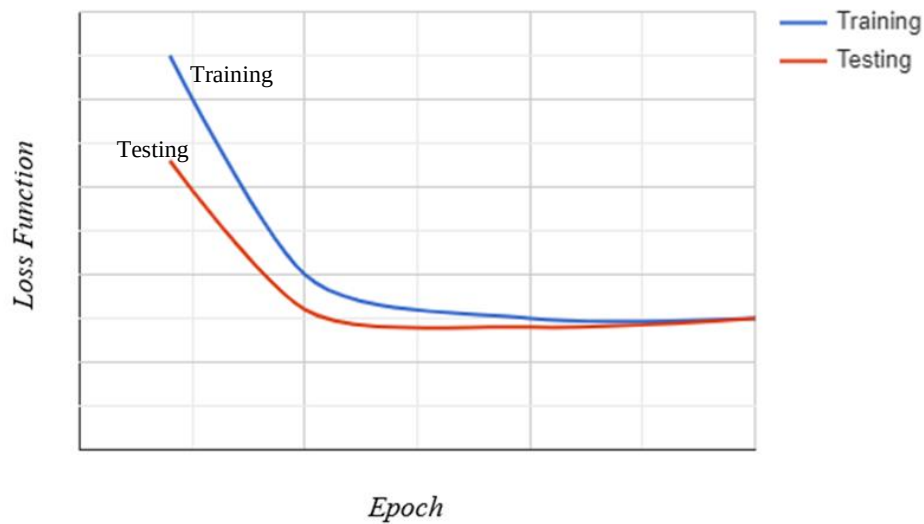


Figure 3.5: The loss curve when the model well-learned

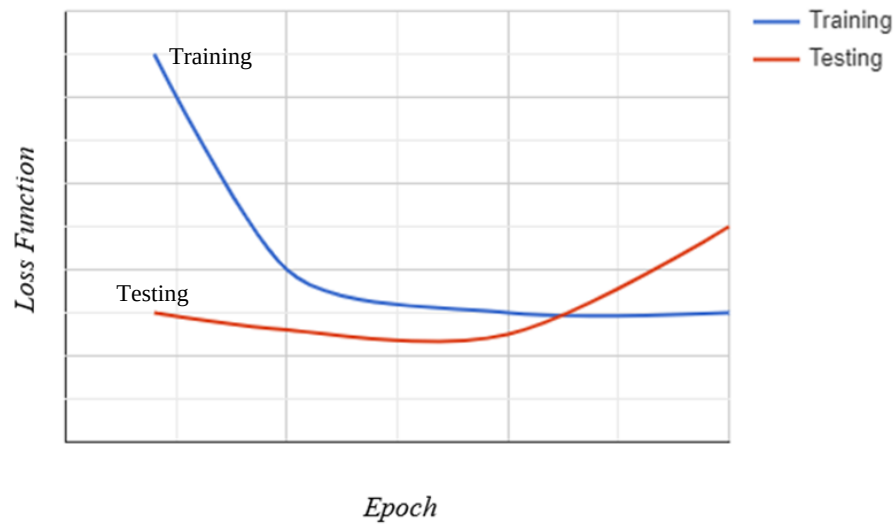


Figure 3.6: The loss curve when the model over-fit

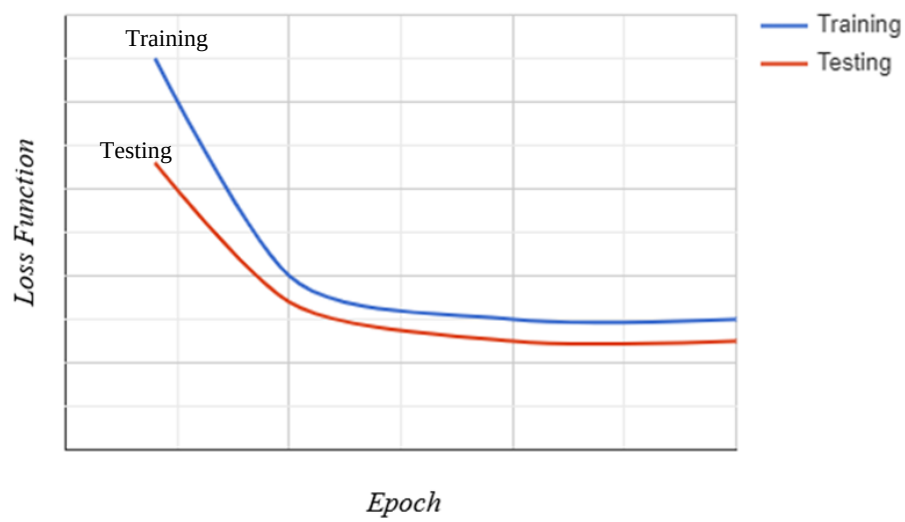


Figure 3.7: The loss curve when the model under-fit

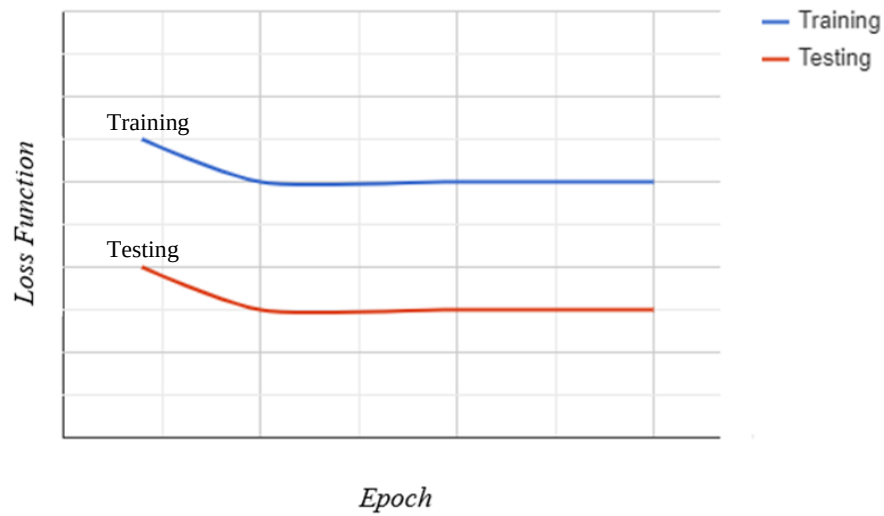


Figure 3.8: The loss curve when the model unable to learn

After decided an appropriate range of epoch value based on the curve pattern similar to Figure 3.5, the training is executed, and the initial result is recorded. The initial result is used as a benchmark to compare with the result of the trained model after the enhancement. Only then, the parameter that was selected under the loss function hyperparameter is tuned to improve the model performance.

For the tuning part, this method allows the developer to make an adjustment to the environment of the neural network according to the initial training result. The parameters usually already have a certain range of values that works optimally with any model. Hence, in this case, the model is trained with the configuration that is defined from that specific range of parameter's value to find the value that can yield the most significant improvement in the result's accuracy from the initial result.

3.4 Summary

In this chapter, the proposed methodology was described. The research flow was first presented, followed by the presentation of crucial steps in building a DNN chatbot model. The research flow consists of five important phases that includes the formulating solution phase, designing proposed enhanced mechanism and hyperparameter selection, implementation, analysis and evaluation, and lastly the documentation phase. The explanation in building the DNN chatbot model discussed the main components of the research, basically emphasizing the importance of dataset pre-processing in the chatbot model, the component defined in the model. This subsection also discusses the implementation of the scaling factor in the Additive attention mechanism as part of the proposed enhancement to scale the dimension of the decoder hidden state. In selecting the parameters in the chosen hyperparameter, the Optimizer, Learning Rate, and the Dropout considered having the highest impact in the training model to output a significant improvement. This chapter also mentioned the model evaluation's metrics through the loss function graph representation and confusion matrix method measured from one-phrase question test. In the next chapter, the results obtained from every different experiment's configuration will be analysed and discussed.

CHAPTER 4

RESULT AND ANALYSIS

4.1 Overview

In the previous chapter, the architectural concepts of the DNN chatbot model and the detailed framework was presented and discussed in detail. In this chapter, the experiment implementation is explained. The training configuration is first explained, followed by the experimental setup and the result with extensive analysis on the obtained result.

4.2 Training Configuration

The Seq2Seq base model was modified in the part of the attention mechanism and executed to conduct model modification and hyper-parameter optimization. The training was done using an open-source hosted Jupyter notebook called Google Colaboratory or Colab in short. Colab is a Google Research product that is specifically designed to work with data analysis and deep learning training. It can be accessed through a browser that needs no configuration to use while providing free access to GPUs.

Besides, Colab is using Python as its scientific programming language, which is one of the most developer-friendly programming languages for Machine Learning and Deep Learning. Python emphasizes the flexibilities and features that enable programmers to improve code quality and their productivity in working with the model while offering an extensive and wide range of libraries such as TensorFlow, Keras, and Scikit-learn which are fundamental in the development requirement.

TensorFlow is one of the open-source Machine Learning libraries that was designed by Google for research and development using data flow diagrams for high-performance

numerical computing. TensorFlow enables users to create graphs of data flows that simulate how data passes through a series of processing nodes through a network or model. These nodes represent a mathematical operation, and the connecting flow is called a multidimensional array that is also another name for Tensor. TensorFlow would also calculate the accuracy and loss function through its various libraries and also known as fast, modular, and scalable.

Meanwhile, Keras is a high-levelled neural networks library, work in parallel with TensorFlow. Keras has a variety of building blocks and resources required to construct a neural network, such as Neural layers, Dropout, Activation and cost functions, and Batch normalization. In comparison to writing a network block of code for a layer in a traditional way, Keras is extremely simple to implement because each layer in the network can be represented with one line of code.

Another machine learning library for Python that is actively used is Scikit-learn. It involves simple integration with various libraries of ML programming, including NumPy and Pandas. Scikit-learn is supported by different algorithms such as Model Selection which may include hyper-parameter tuning (the focus of current work), Evaluation that include the metrics and validation curves, and Dataset Transformation.

The overall training was sufficiently done on a Windows system that was running the NVIDIA GeForce MXX330 with a graphic clock at 1531 MHz. The CPU used is 11th Gen Intel(R) Core (TM) i7 @ 2.80GHz that has 8GB RAM.

4.3 Experimental Setup

In Chapter 3, the guidelines are given to find the suitable epoch for the training in every configuration through the illustration of the training's graph. Later, in all of the experiments, the final accuracy was recorded on the optimal value of the epoch according to the guideline from the learning curve's graph (Figure 3.5). The reason is that, every configuration has a different environment that requires a specific value of epoch. But, in this research, epochs at 10 is recorded optimum for all this research.

For an important remark, the highest accuracy is indicated by the lowest loss rate. Through the rest of the experiment, the model performance will be measured on its accuracy rate while taking note of good progress of declination in the loss rate.

There are total of six experiments conducted in this Chapter. The experiment first done on the comparison between basic Attentive Seq2Seq model and Enhanced Attention Seq2Seq model, described in Section 4.4. Then, series of experiments for optimizing the capacity of training's network executed which is described in Section 4.5. It is included the experiments of encoder type selection, comparison on performance of optimizer's types, learning rate and dropout. The last experiment in Section 4.6 executed on the comparison of three different model which are the basic Attentive Seq2Seq model, Enhanced Attention Seq2Seq model, and Enhanced Attention Seq2Seq model with optimized network.

4.4 Implementing the Enhanced Attention (EA)

The objective of this experiment is to examine the performance of proposed enhanced attention after adding in the baseline attentive Seq2Seq model. The method used in this experiment is by running a model with an enhanced attention mechanism according to the modified algorithm - Refer pseudocode in Chapter 3 (Figure 3.3).

Table 4.1 below shows a model configuration for the baseline attentive Seq2Seq model.

Table 4.1: Proposed enhancement experimental configuration

Model Configuration	Value
Optimizer	Adam
Percentage of validation split	0.1%
Batch Size	128
Dropout	0.05
Epoch	10
Encoder Type	Uni-directional

In this experiment, two models that include a baseline model (Attentive Seq2seq) and an enhanced model (Enhanced Attentive Seq2Seq) will be trained simultaneously using the configurations as defined by default above. Table 4.2 presented a result obtained after training, while Figure 4.1 below illustrated the comparison result with the y-axis indicating the accuracy rate and the x-axis showing the number of epochs.

Table 4.2: Accuracy rate Attentive Seq2Seq and EA Seq2Seq

Epoch	Attentive Seq2Seq	EA Seq2Seq
1	0.751447	0.749303
2	0.777889	0.777349
3	0.783025	0.782703
4	0.785196	0.785892
5	0.787285	0.788211
6	0.788939	0.790127
7	0.790389	0.791407
8	0.79156	0.792976
9	0.793001	0.797562
10	0.794198	0.810946

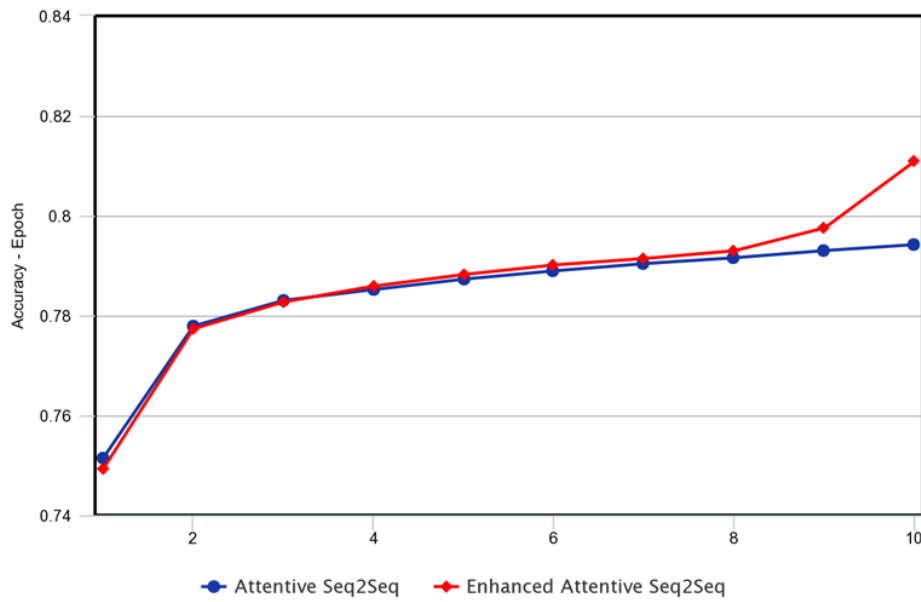


Figure 4.1: Accuracy performance Attentive Seq2Seq vs EA Seq2Seq

The result shows the final accuracy obtained through the training after implementing a slight modification in the algorithm is at 81% accuracy rate compared to the basic model that recorded its final accuracy at 79% accuracy rate.

To justify the results, as mentioned in Subsection 3.3.3 (proposed enhancement design), a scaling factor is added to scale the dimension of the decoder hidden state. The scale function is to create stability between the sum of addition vector input and the Softmax function. The reason is, when the vector is executed in softmax function, it resulted a small value in which it sums up the vector values to 1 over the whole sequence. The range of individual value lies between 0 to 1.

This number is considered relatively small compared to the input value that growing large in magnitude by the nature of exponential function. This may cause an unbalanced gradient between the value of input and softmax output. Therefore, when the scaling factor is added, it helps to counteract the problem by achieving more stable gradients, hence leads the Softmax score to determine the priority of focus in the sentence of intent from users.

Hence, it can be said that this factor successfully contributes an improvement on the training's accuracy regardless it only increased in small value.

4.5 Optimizing the Capacity of Training's Network

4.5.1 Encoder Type Selection

The objective of this experiment is to examine and analyse the performance of two different encoder types in the Seq2Seq model. The model configuration is fixed into these values that tabulated in Table 4.3.

Table 4.3: Encoder type experimental configuration

Model Configuration	Value
Optimizer	Adam
Percentage of validation split	0.1%
Batch Size	128
Dropout	0.05
Epoch	10

In this experiment, two models implemented with a uni-directional encoder and the bi-directional encoder will be trained using the configurations as defined above. Table 4.4 presented a result obtained after training, while Figure 4.2 below illustrated the comparison result with the y-axis indicating the accuracy rate and the x-axis showing the number of epochs.

Table 4.4: Accuracy rate for encoder type

Encoder Type Epoch	Uni-directional	Bi-directional
1	0.74964	0.751854
2	0.776991	0.779165
3	0.782201	0.784968
4	0.785607	0.7882
5	0.787293	0.791289
6	0.789145	0.794268
7	0.790736	0.798287
8	0.792101	0.803759
9	0.793244	0.810546
10	0.794566	0.817987

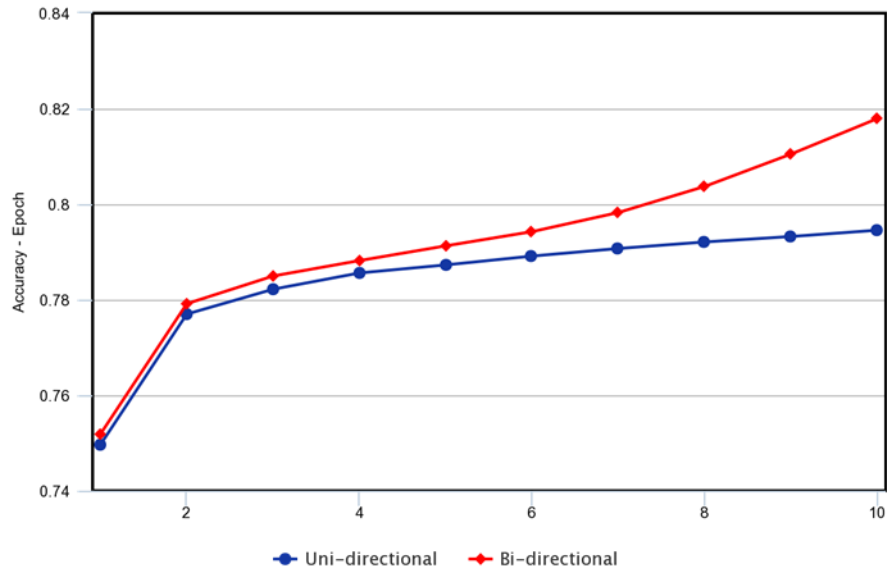


Figure 4.2: Encoder performance Uni-directional vs Bi-directional

The result indicates that the model with a bi-directional encoder performs better than the model with a uni-directional encoder. It is because the bi-directional encoder works by preserving the information from both directions of the sentence, allowing the network to predict the next word better as it can understand the sentence context. Hence, the best encoder type to work with this model is finalized into a bi-directional encoder in the final experiment of optimizing the chatbot model.

4.5.2 Identifying the Best Optimizer

The objective of this experiment is to measure the performance of optimizer according to the model environment used in this research whether the original optimizer used in basic model can also work best with enhanced model. The experimental setup for comparing the best optimizer for chatbot model starts with identifying three most popular optimizer that already has been proven works the best in their own configuration. The three optimizers that will be tested includes AdaGrad, RMSProp, and Adam. Each optimizer is configured with the default hyperparameter value from Tensorflow Library that is shown in Table 4.5.

Table 4.5: Default optimizers' configuration

Optimizer	Configuration Parameters		
AdaGrad	Learning Rate = 0.001	Initial Accumulator = 0.1	Epsilon = 1e-7
RMSProp	Learning Rate = 0.001	Rho = 0.1	Epsilon = 1e-7
Adam	Learning Rate = 0.001	Beta = 0.9 & 0.999	Epsilon = 1e-7

Based on the model configuration that is automatically defined in Tensorflow as stated in Table 4.6, the experiment is conducted by using three different optimizers, which includes AdaGrad optimizer, Adam, and RMSProp optimizer. Table 4.7 presents a result

obtained after training, while Figure 4.3 shows a line graph on the comparison result with the y-axis indicating the accuracy rate and the x-axis showing the number of epochs.

Table 4.6: Optimizers experimental configuration

Model Configuration	Value
Batch Size	128
Percentage of validation split	0.1%
Dropout	0.05
Epoch	10
Encoder Type	Bi-directional

Table 4.7: Accuracy rate of optimizer types

Optimizer Epoch	AdaGrad	Adam	RMSProp
1	0.648795	0.756129	0.751552
2	0.68112	0.781034	0.778956
3	0.68112	0.786772	0.784894
4	0.68112	0.790331	0.788547
5	0.681137	0.793363	0.791664
6	0.682003	0.797029	0.794483
7	0.687645	0.80147	0.799118
8	0.6968	0.806918	0.805488
9	0.704754	0.813169	0.813078
10	0.710098	0.821428	0.821223

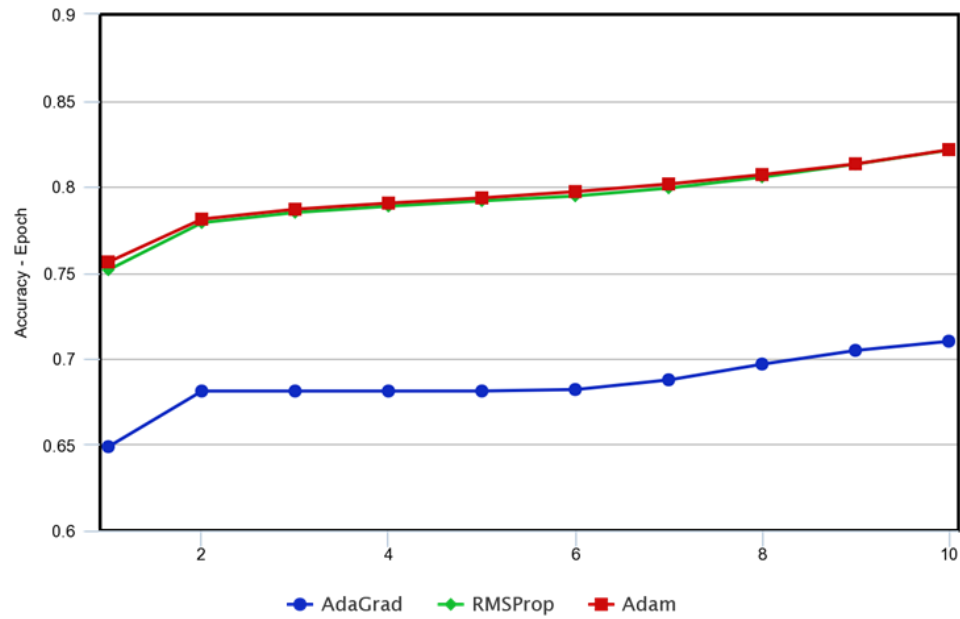


Figure 4.3: Accuracy performance of optimizer types

Based on these results, Adam optimizer should be used as an optimizer in the Seq2Seq chatbot model since it got the highest training accuracy. Since it is not much difference in the performance of accuracy between Adam and RMSprop optimizer, the loss rate value is also measured as an indicator to find the best optimizer. To emphasize again, the lower the loss rate value, the better the performance is. Table 4.8 shows the final value of accuracy and loss rate for all optimizers.

Table 4.8: Final accuracy rate and loss rate of optimizers

Optimizer	Training Accuracy	Training Loss
AdaGrad	0.7100	2.0535
RMSProp	0.8212	0.8646
Adam	0.8214	0.7900

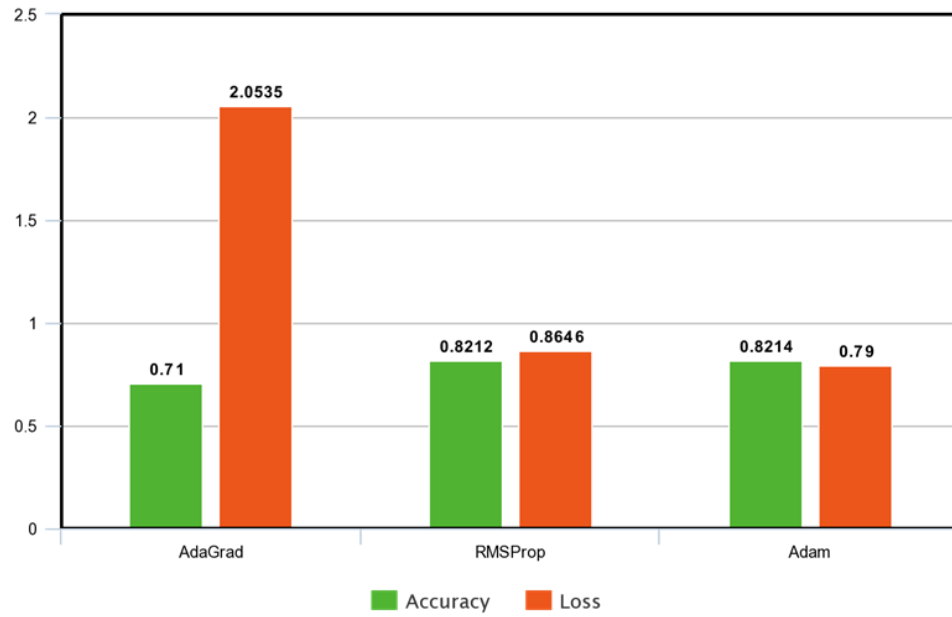


Figure 4.4: Model performance according to optimizer types

In Figure 4.4 of the bar chart that depicts the comparison of accuracy value and loss rate between the three optimizers, Adam optimizer yields the lowest loss rate value compared to the others optimizers trained during the experiment.

A good neural network model should maximize the loss, so it should be low as possible. So, an optimizer should be able to make the model learning converge adequately in reducing the loss where the model can handle sparse gradients on noisy problems. In addition, to select the best optimizer for the model, the criteria needed for the optimizer is, it is capable of generating a high accuracy value and low loss rate value.

Therefore, by considering both accuracy value and the loss rate value, Adam optimizer achieves both criteria and can works better for the model compared to RMSprop.

4.5.3 Fine Tune Learning Rate

In the previous section, the type of optimizers has been discussed on which one works better, and it is proven that Adam optimizer outperforms other optimizers. From the experiment, we can see every optimizer has its own configuration, and remarkably each of them has a learning rate parameter. The optimizers have already set the learning rate into default value which is considered just to work moderately enough for the model to perform well.

For this experiment, the learning rate will be tuned to optimize the optimizer to find if the enhancement of the optimizer's performance could also increase the model's accuracy. To see the performance variance of the learning rate, the values will be set on a logarithmic scale from 0.0001 for lower bound to 1.0 for the upper bound. The model configuration except learning rate is fixed into these values that can be seen in Table 4.9.

Table 4.9: Learning rate experimental configuration

Model Configuration	Value
Learning rate	0.0001, 0.001, 0.01, 0.1, 1.0
Optimizer	Adam
Percentage of validation split	0.1%
Batch Size	128
Dropout	0.05
Epoch	10
Encoder Type	Bi-directional

The result will consist of 5 plots that measure the model's accuracy based on different values of learning rate that is recorded in Table 4.10. Figure 4.5 then will show a line graph

on the comparison result with the y-axis indicating the accuracy rate and the x-axis showing the number of epochs.

Table 4.10: Accuracy rate of learning rate parameter

Learning Rate Epoch	0.0001	0.001	0.01	0.1	1.0
1	0.711752	0.752959	0.765149	0.730036	0.699267
2	0.752739	0.779535	0.789297	0.768356	0.726783
3	0.761717	0.785873	0.794	0.774651	0.730083
4	0.769516	0.789388	0.799243	0.776765	0.731567
5	0.773661	0.792854	0.806967	0.777645	0.733217
6	0.775269	0.797012	0.817847	0.777442	0.734326
7	0.77645	0.802418	0.829573	0.77717	0.734675
8	0.77757	0.809639	0.84108	0.777683	0.73555
9	0.778826	0.818822	0.851131	0.777857	0.73526
10	0.77968	0.829064	0.859454	0.777987	0.735475

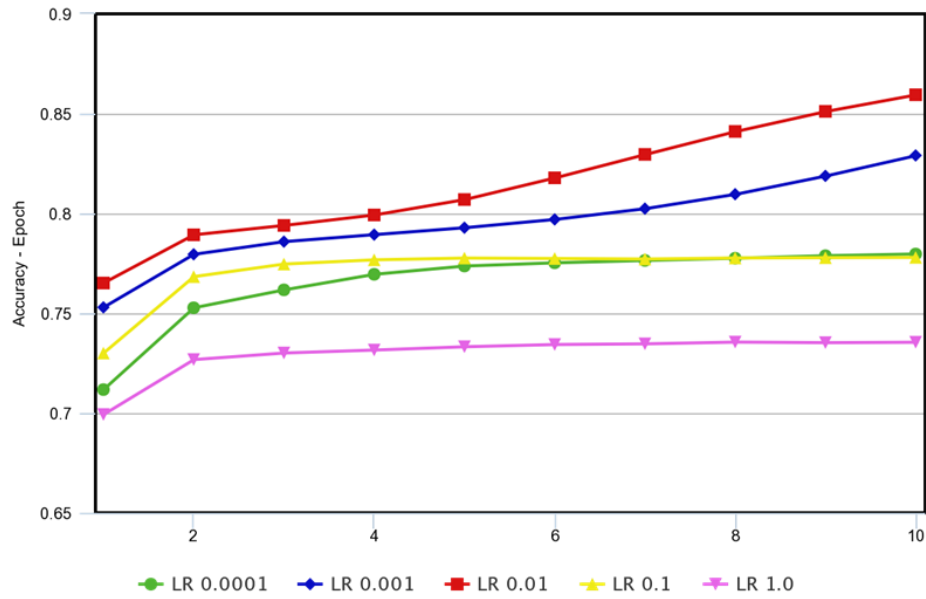


Figure 4.5: Accuracy performance of the value learning rate used

The plot shows the behaviour of the learning rate affecting the model's learning. Learning rate with the largest value (1.0) depicting the erratic behaviour going back and forth, converging quickly, making the model unable to learn, resulting in the lowest accuracy rate. Meanwhile, the learning rate with a value (0.1) made the model unable to learn efficiently. The line graph of value (0.1 & 1.0) that shows a straight line indicates the model get stuck in the learning process.

To highlight the point, when the learning rate converging quickly (gradient was updated too frequent), the model distortion would happen and affecting the model ability to get over local minima. The constant line indicates that the gradient is infrequently updated or has not been updated at all.

Eliminating the worst value, the learning rate of 0.0001, 0.001, and 0.01 depicts the stable graphs, which show a normal learning activity. But in comparison, a learning rate of 0.01 can be seen to work well with the model, whereas it can learn the problem adequately,

yielding a final accuracy at 85% accuracy rate. Hence, the value of moderate learning rate is finalized into 0.01 as the best value of learning rate to work with Adam optimizer in the final experiment of optimizing the chatbot model.

4.5.4 Fine Tune Dropout

Dropout is implemented per layer in a neural network. In this study, the dropout is implemented in recurrent layers that include hidden layers as well the visible or input layer. Dropout will make the weights of the network will be larger than usual. Therefore, before finalizing the network, the weight will be scaled first by the dropout rate, reducing the error rate. On the other note, in a recurrent neural network (RNN), a dropout is applied only to feed-forward connections.

In the neural network setting, the value of 1.0 means no dropout and 0.0 means no output from the layer. For the experiment, the dropout will be set on a scale of 0.05 from the lower bound to 0.1 for the upper bound. The model configuration as in Table 4.11 except dropout is fixed into these values:

Table 4.11: Dropout experimental configuration

Model Configuration	Value
Dropout	0.05, 0.06, 0.07, 0.08, 0.09, 0.1
Learning rate	0.01
Optimizer	Adam
Percentage of validation split	0.1%
Batch Size	128
Epoch	10
Encoder Type	Bi-directional

The experiment starts off with training the baseline model that was implemented with the dropout of 0.05. After that, the experiment continues by adding dropout values from 0.05 to 0.1 for every session, and the results are recorded in Table 4.12 and visualized as below in Figure 4.6 with the y-axis indicating the accuracy rate and the x-axis showing the number of epochs.

Table 4.12: Accuracy rate of dropout parameter

Dropout Epoch	0.05	0.06	0.07	0.08	0.09	0.1
1	0.765149	0.772109	0.770538	0.767581	0.76708	0.762688
2	0.789297	0.790423	0.790657	0.789735	0.789056	0.788038
3	0.794	0.795439	0.796171	0.794306	0.793101	0.791981
4	0.799243	0.802079	0.803231	0.800617	0.796797	0.794766
5	0.806967	0.811792	0.815315	0.810625	0.80226	0.798898
6	0.817847	0.824868	0.829305	0.823199	0.8087	0.80447
7	0.829573	0.838707	0.842386	0.835769	0.815758	0.810263
8	0.84108	0.850944	0.853384	0.847186	0.822352	0.815288
9	0.851131	0.860759	0.86282	0.856211	0.828724	0.820001
10	0.859454	0.869341	0.870544	0.864344	0.833803	0.824385

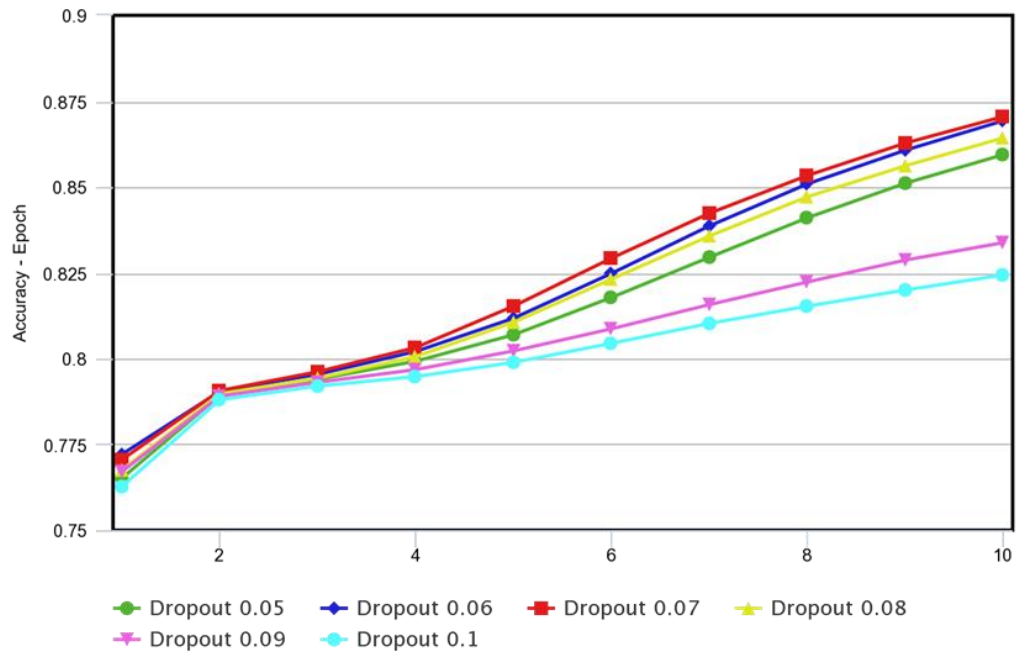


Figure 4.6: Accuracy performance of dropout value used

To give a clear explanation, the graph as in Figure 4.7, y-axis that indicates the loss rate, and the x-axis showing the number of epochs illustrated the comparison of the loss rate value after applying these dropout values.

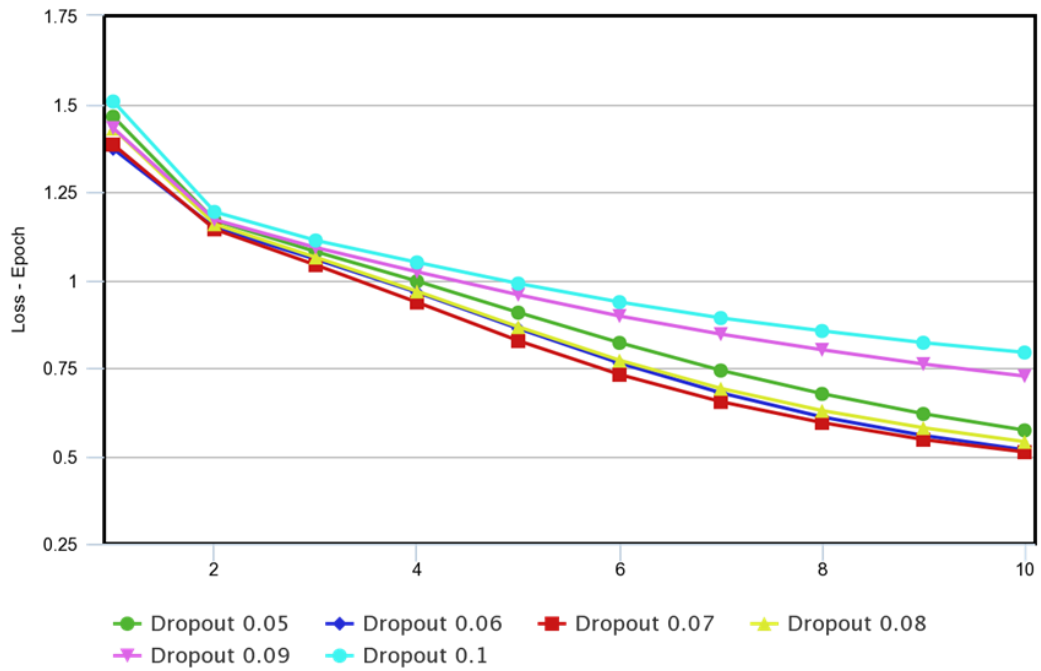


Figure 4.7: Loss rate according to dropout values used

The graph shows the loss curve (error rate) according to the respective dropout rate. The value between 0.06 and 0.07 indicates that the dropout successfully reducing the error rate by extending from overfitting that led to the maximum regularization, resulting in the accuracy rate recorded at 86% and 87%, respectively.

The upper bound value of 0.1 is too large for the model beyond the optimal dropout threshold, which resulting in a higher error rate. Meanwhile, the value of the lower bound at 0.05 slows down the convergence, and the model unable to fit properly.

In conclusion, the dropout value that suits the best to be used in the final experiment is value of 0.07.

4.6 Baseline Model vs Optimized Model

In this final experiment, all of the final findings recorded from the previous experiment will be used from here on. The objective of the experiment is to finalize the theory and actual findings into one final conclusion through data representation that will be illustrated from the result of the model's training.

The experiment involves training three models, which are the Attentive Seq2Seq model (baseline model), the Enhanced Attentive Seq2Seq model (implemented with proposed algorithm modification) and Enhanced Attentive Seq2Seq with hyperparameter optimization. These models will be trained based on their specific setting that would define how the model works.

Table 4.13 below shows the summary of the finding for the previous experiment. It also serves as the baseline configuration and the final configuration after the optimal value and type selection for this experiment.

Table 4.13: Final chatbot performance experimental configuration

Hyperparameter	Baseline	Range Value		Optimal Value
		Lower bound	Upper bound	
Optimizer	Adam	-		Adam
Learning rate	0.001	0.0001	1.0	0.01
Dropout	0.05	0.05	0.1	0.07
Encoder Type	Uni-directional	-		Bi-directional

The models were then trained on Cornell Movie-Dialogue Corpus, run by using 10 epochs to compare the model performance according to the accuracy and loss function. The

results are tabulated in Table 4.14 and illustrated as in Figure 4.8. The graph indicates the comparison of accuracy value and loss rate between the three models.

Table 4.14: Comparison result of overall model performance

Model	Loss	Accuracy
Attentive Seq2Seq (Baseline)	1.051316	0.794198
Attentive Seq2Seq + EA	1.011107	0.810946
Seq2Seq + EA + Optimized network	0.510229	0.870544

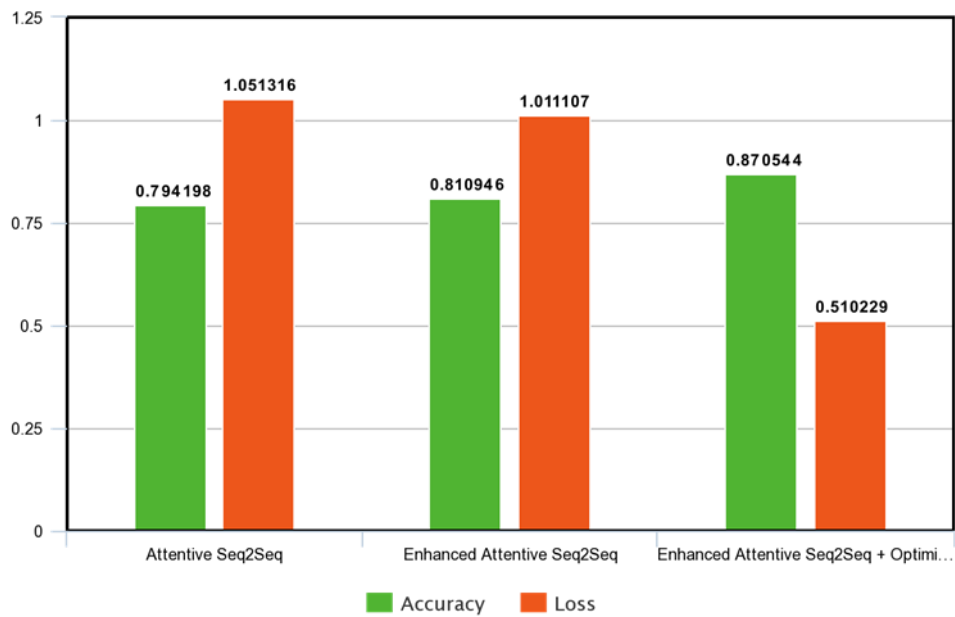


Figure 4.8: Overall model performance

The graph shows that there is a significant improvement on model's performance after modification and optimization that recorded its final value of accuracy and loss rate at 87% and 0.51 respectively compared to the result before implementing the proposed

modification (79% accuracy and 1.05 loss), and after implementing the proposed enhancement but without hyperparameter optimization (81% accuracy and 1.01 loss).

This result undoubtedly indicates the optimized model outperforms the baseline model.

4.7 Evaluation

In this section, the method of a confusion matrix is applied to calculate the precision of actual response Attentive Seq2Seq model (baseline model) and Enhanced Attentive Seq2Seq with hyperparameter optimization (enhanced model).

Before that, a one-phrase question test is conducted first, and the result is recorded in Table 4.15. In addition, the precision calculation will be based on the matrix labelled for every question, either True Positive (TP) or False Positive (FP).

A collection of questions is based on a random topic without any boundary of a knowledge base but also referenced from existing experiment from the works of Csaky, (2019). There are 10 questions are presented based on the answer labelled as the matrix of TP and FP only.

Table 4.15: One-Phrase Question Test

Question	Expected answer	Actual Response		Matrix	
	Baseline model	Baseline model	Enhanced model	Baseline model	Enhanced model
Hello!	Hello, Hi	Hello.	Hi	TP	TP
How are you?	Fine.	Fine.	Fine. How are you?	TP	TP
What is your name?	<i>Any name</i>	My name is sir robin of Camelot!	David.	TP	TP
What is your job?	<i>Any job</i>	I like working	I like working	FP	FP
What is the colour of the sky?	Blue	I don't know	White	FP	FP
Ok bye!	Bye!	Bye!	Bye!	TP	TP
Good morning	Good morning	Morning	Good morning	TP	TP
What is your favourite colour?	<i>Any colour</i>	There are seven colours in a rainbow	Red is my favourite colour.	FP	TP
What are you doing?	<i>Any verb</i>	I'm playing	Sleeping	TP	TP
What is your favourite food?	<i>Any food</i>	Burger	Burger	TP	TP

Precision of actual response for: = $TP / (TP + FP)$

$$\begin{aligned}\textbf{Baseline model:} \quad \text{Precision} &= 7 / (7 + 3) \\ &= 0.7\end{aligned}$$

$$\begin{aligned}\textbf{Enhanced model:} \quad \text{Precision} &= 8 / (8 + 2) \\ &= 0.8\end{aligned}$$

The result shows that the precision value in the one-phrase question test for the baseline model is 0.7 while for the enhanced model is 0.8. This indicates that an enhanced model with the implementation of algorithm modification and hyperparameter optimization outperforms an existing Attentive Seq2Seq model and also become a proof that the training result is reliable because of the same small improvement in value of Precision.

4.8 Discussion

The experiments conducted in this chapter all was done to achieve the objectives presented in Chapter 1. The first objective is to propose an optimization strategy based on Structural Modification and Optimizing Training Network for improving the lacking of accuracy of response in the chatbot application. To achieve this objective, a methodology was established in identifying the important components that can be modified in the Additive Attention algorithm. The crucial hyperparameter followed by high impacts parameters also has been chosen to be measured on its performance in the experimental section.

After identifying the strategies, objective number two then achieved by proposing the algorithm enhancement to improve the current attention mechanism in the Attentive Sequence-to-Sequence model and the network's training optimization because of its inability to memorize the dialogue history. The Additive attention algorithm was modified by adding scale factor between decoder hidden state and context vector. The result of the training shows that the enhanced model with the modified algorithm of attention mechanism slightly

outperforms the basic Attentive Seq2Seq model. The finding proves that the addition scaling factor successfully help the model determines the priority of focus in the sentence of intent from users because of the stable gradient in attention scores.

After that, the experiments to achieve the second objective continued by the network optimization through the experiment with which encoder type works the best for the model, experimenting different types of optimizers, a range of learning rate and dropout values to find the most optimal one. The result shows that bi-directional encoders yield high accuracy compares to a uni-directional encoders on the factor of the ability to pinpoint any information that is relevant from the sentence. Besides, between the optimizers of AdaGrad, RMSProp and Adam, Adam optimizer successfully make the model learning converge adequately in reducing the loss where the model can handle sparse gradients on noisy problems, resulting high accuracy and low loss value.

The experiment in finding optimal learning rate ended with the value 0.01 as the most optimal one through the accuracy value yield by the model. It indicates that learning rate on 0.01 value capable in making the model learn adequately and work well with Adam optimizer. Then, the experiment to find out optimal value for dropout was carried out and the result shows that the value of 0.07 is the optimal value and indicates that the dropout successfully reducing the error rate by extending from overfitting that led to the maximum regularization.

The experiment ends with comparing the model's result after training with the basic Attentive Seq2Seq model, Enhanced Attentive Seq2Seq model, and enhanced model with hyperparameter optimization using the final optimum value respectively. The result shows that the enhanced model with hyperparameter optimization performed better than the other two. Hence, after getting the final accuracy value for the proposed enhanced model, the

model was evaluated to prove the precision of accuracy value for the trained model which is cater the third objective which is to evaluate the accuracy of response of the proposed solution through data training on loss function and real data testing. The one-phrase question test was conducted and the result shows that the precision value in the one-phrase question test for the enhanced model is 0.8. This become a proof that the training result is reliable because of the same small improvement in value of Precision.

4.9 Chapter Summary

In this chapter, the result and analysis of the proposed enhancement are presented. Before that, a training configuration was first laid down, followed by a brief guideline on the experimental setup. Equipped with the guideline given, the first experiment was conducted on training a model with an enhanced attention mechanism. The result of the training shows that the enhanced model with the modified algorithm of attention mechanism slightly outperforms the basic Attentive Seq2Seq model. The finding proves that the addition scaling factor successfully help the model determines the priority of focus in the sentence of intent from users because of the stable gradient in attention scores. After that, the experiments continue by experimenting with which encoder type works the best for the model. The result shows that bi-directional encoders yield high accuracy compares to a uni-directional encoders on the factor of the ability to pinpoint any information that is relevant from the sentence. This chapter also discussed which and why the value is optimal for every parameter selected. The values are finalized as a configuration for the final experiment. The experiment ends with comparing the model's result after training with the basic Attentive Seq2Seq model, Enhanced Attentive Seq2Seq model, and enhanced model with hyperparameter optimization. The result shows that the enhanced model with hyperparameter optimization performed better than the other two. The chapter wrapped up with an evaluation of the

baseline model and enhanced model to validate all of the results obtained through the experiments. In conclusion, the enhanced model with the implementation of algorithm modification and hyperparameter optimization outperforms an existing Attentive Seq2Seq model, not only in the accuracy value in model training but also in the evaluation of model testing using real test data.

CHAPTER 5

CONCLUSION

In this chapter, a summary of research contributions is being concluded, followed by a discussion on the future research works and a final conclusion, to sum up this research.

5.1 Summary of Research Contributions

In this study, there are three research objectives that were established and discussed in Chapter 1. The main objective is to optimize the capability of the chatbot model in capturing the focus of user intent in the query sentence and optimizing the model network training's capacity. In order to achieve the main objectives, the three specific research objectives were laid down. The first one was to propose the algorithm enhancement to improve the current attention mechanism in the Seq2Seq model. The second research objective was to propose an optimization strategy for improving the accuracy of response for a chatbot. The last objective was to evaluate the proposed solution through data training on loss function and confusion matrix.

Towards the end of this study, the following research contributions were achieved on enhancement strategy by structural modification and the optimization of the network training's environment in the DNN chatbot model:

- a) Structural modification – a slight modification as an enhancement in the Additive Attention Mechanism model by scaling the dimension of the decoder hidden state. The purpose is to balance the input size and Softmax function.
- b) Network training's environment optimization through a hyperparameter optimization. The loss function is selected as high impact hyperparameter while

fine-tuning the most contributing parameters to the network training, which are the Optimizer, Learning Rate, and Dropout.

The strategy adoption has proven can improve both training accuracy and the quality of answer output by the model.

5.2 Conclusion and Future Works

Throughout the research, a lot of existing works linked to the DNN chatbot model are reviewed that lead to the conclusion that the DNN chatbot model still does not achieve its highest capability. Hence, the first step for the research is to identify the best architecture that works with the chatbot. Seq2Seq architecture already proven can tackle a chatbot problem setting better than the other models. After identifying the architecture, it is important to decide how to improve the model.

The very first idea of this research is to adopt an improvement strategy to the existing model, as in any way, it can open up to a lot of potential of another enhancement method. Inspired from many reviewed works, it is decided that the enhancement in this research can be categorized into two categories, including the structural modification and the optimization of the network training's environment, as both can contribute a significant improvement to the model.

The structural modification in this research focusing on the attention mechanism where there is a slight modification in the algorithm of the Additive attention mechanism. Meanwhile, the network training's environment is optimized by deciding first the high impact hyperparameters – the loss function, and then fine-tuned the selected parameters 'value (Optimizer, Learning Rate, and Dropout).

The final result after implementing the proposed enhancement as compared to the current Attentive Seq2Seq model. The comparison between the models showed that the Enhanced Attentive Seq2Seq model with hyperparameter optimization performed better than the baseline model both in training's accuracy and model's evaluation using real test data. Although the enhanced model only made a slight improvement in the training accuracy value and the precision value against the baseline model, the results of this work showed that the strategy used in improving the model's performance contains research opportunities to increase the chatbot performance.

This study has the potential to be extended into the full-scale works. Therefore, some future works can be worked on with sufficient time and resources. There are two main factors that can be added into the future works, which suggesting the method used in hyperparameter optimization and enhancement of another component in the structural modification. The first future work is fine-tuning the hyperparameter by automatic value selection. The traditional method used in this study actually is time-consuming and expensive. By using the automatic value selection method, the fine-tuning part in the hyperparameter optimization will be less expensive and will likely identify the optimal value more accurate than the current method. The second future work has proposed an enhancement in part of structural modification by adding embedding and additional encoder. The enhancement in other components too can help the model to learn better and achieve a high understanding of the sentence's context. By having the model's improvement in problem learning capability and language understanding, the accuracy of the model can be improved by major scale.

REFERENCES

- Agnihotri, S. (2019). *Hyperparameter Optimization on Neural Machine Translation*. [e-book] Ames: Iowa State University Digital Press. Available through: Iowa State University of Science and Technology Digital Repository
- Andrew, S., Georg, H., Marc'Aurelio, R., & Ke, Y. (2013). An empirical study of learning rates in deep neural networks for speech recognition. *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, 1, 6724-6728.
- Ashgar, N., Poupart, P., Hoey, J., Jiang, X., & Mou, L. (2018). Affective neural response generation. *Proceeding European Conference Information Retrieval*, 154-166.
- Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural Machine Translation by Jointly Learning to Align and Translate. *Proceeding International Conference on Learning Representations*, 1-15.
- Bansal, H., & Khan, R. (2018). A Review Paper on Human Computer Interaction. *International Journals of Advanced Research in Computer Science and Software Engineering*, 8(4), 53-56.
- Caldarini, G., Jaf, S., & McGarry, K. (2022). A Literature Survey of Recent Advances in Chatbots. *Information*, 41.
- Cho, K., van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. *Empirical Methods Natural Language Processing (EMNLP)*, 1724-1734.
- Csaky, R. K. (2019). Deep Learning Based Chatbot Models. *arXiv*.
- Das, S., & Kumar, E. (2018). Determining Accuracy of Chatbot by Applying Algorithm

- Design and Defined Process. *International Conference on Computing Communication and Automation (ICCCA)*.
- Deng, L., & Dong, Y. (2014). Deep Learning: Methods and Applications. *Found. Trends Signal Process*, 7, 197-387.
- Elbayad, M. (2020). Rethinking the Design of Sequence-to-Sequence Models for Efficient Machine Translation. *Machine Learning*.
- Elgeldawi, E., Sayed, A., Galal, A. R., & Zaki, A. M. (2021). Hyperparameter Tuning for Machine Learning Algorithms. *Informatics 2021*, 8(79).
- Galassi, A., Lippi, M., & Torroni, P. (2020). Attention in Natural Language Processing. *IEEE Transactions on Neural Networks and Learning Systems*.
- Genthial, G. (2017, November 8). *Seq2Seq with Attention and Beam Search*. Retrieved from Guillaume Genthial blog: <https://guilleaumegenthial.github.io/sequence-to-sequence.html>
- Grossi, E., & Buscema, M. (2008, January). Introduction to artificial neural networks. *European Journal of Gastroenterology & Hepatology*, 1046 - 1054.
- H. Almansor, E., & Hussain, F. K. (2020). Survey on Intelligent Chatbots: State-of-the-Art and Future Research Directions. *Springer Nature Switzerland AG*, 534 - 543.
- Hajiabadi, M., Farhadi, M., Babaiyan, V., & Estebarsari, A. (2020, August 7). Deep Learning with Loss Ensembles for Solar Power Prediction in Smart Cities. *Smart Cities*, 3, 842–852.
- Hardalov, M., Koychev, I., & Nakov, P. (2018). Towards Automated Customer Support. *Proceedings International Conference Artificial Intelligence Methodology System Application*, 48-59.
- Hien, H. T., Cuong , N. P., Nam , H. L., Ho , N. L., & Thang , D. L. (2018). Intelligent

- Assistants in Higher-Education Environments: The FIT-EBot, a Chatbot for Administrative and Learning Support. *The 9th International Symposium On Information And Communication Technology*, 69-76
- Hochreiter, S. (1998). The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 107-116.
- Hu, Y., Huber, A., Anumula, J., & Liu, S.-C. (2018). Overcoming the vanishing gradient problem in plain recurrent networks. *ArXiv*.
- Ketkar, N. (2017). Stochastic Gradient Descent. In N. Ketkar, *Deep Learning with Python* (pp. 111-130).
- Kingma, D. P., & Ba, J. (2015). Adam: A Method for Stochastic Optimization. *Computing Research Repository*.
- Li, J., Galley, M., Brockett, C., Spithourakis, G., Gao, J., & Dolan, B. (2016). persona-based neural conversation model. *Proceedings of the 54th Annual Meeting Association Computational Linguistics*, 994-1003.
- Li, M., Zhang, T., Chen, Y., & Smola, A. (2014). Efficient mini-batch training for stochastic optimization. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*.
- Lydia, A., & Francis, S. (2019). Adagrad - An Optimizer for Stochastic Gradient Descent. 566-568.
- Mallios, S., & Bourbakis, N. (2016). A Survey on Human Machine Dialogue Systems. 1-7.
- Mou, L., Song, Y., Yan, R., Li, G., Zhang, L., & Jin, Z. (2016). Sequence to backward and forward sequences: A content-introducing approach to generative short-text conversation. *Proceedings International Conference on Computational Linguistics*,

3349-3358.

- Mukkamala, C. M., & Hein, M. (2017). Variants of RMSProp and Adagrad with Logarithmic Regret Bounds. *In Proceedings of the 34th International Conference on Machine Learning*, 70, 2545–2553.
- Nimavat, K., & Champaneria, T. (2017). Chatbots: An overview. Types, Architecture, Tools and Future Possibilities. *International Journal Scientific Research & Development*. Ahmedabab.
- Nuruzzaman, M., & Hussain, O. K. (2018). A Survey on Chatbot Implementation in Customer Service Industry through Deep Neural Networks. *IEEE 15th International Conference on e-Business Engineering (ICEBE)*, 54-61.
- Olah, C. (2015, August 27). *Understanding LSTM Networks*. Retrieved from colah's blog: <https://colah.github.io/>
- Paikari, E., & van der Hock, A. (2018). A Framework for Understanding Chatbots and their Future. *2018 ACM/IEEE 11th International Workshop on Cooperative and Human Aspects of Software Engineering*.
- Palasundram, K., M. Shareef, N., Kasmiran, K. A., & Azman, A. (2020). Enhancements to the Sequence-to-Sequence-Based Natural Answer Generation Models. *IEEE Access*, 8, 45738-45752.
- Pascanu, R., Mikolov, T., & Bengio, Y. (2013). On the difficulty of training Recurrent Neural Networks. *30th International Conference on Machine Learning (ICML)*.
- Prabowo, D. Y., Spits Warnars, H., Budiharto, W., Kistijantoro, A., Heryadi, Y., & Lukas, L. (2018). Lstm And Simple Rnn Comparison In The Problem Of Sequence To Sequence On Conversation Data Using Bahasa Indonesia. *2018 Indonesian Association for Pattern Recognition International Conference (INAPR)*, 51-56.

- Qi, J., Du, J., Siniscalchi, M., & Ma, X. (2020). On Mean Absolute Error for Deep Neural Network Based Vector-to-Vector Regression. *IEEE Signal Processing Letters*.
- Ranoliya, B. R., Raghuwanshi, N., & Singh, S. (2017). Chatbot for university related FAQs. *International Conference on Advances in Computing, Communications and Informatics*, 1525-1530.
- Ruder, S. (2016). An overview of gradient descent optimization algorithms. *ArXiv*.
- Santosh Kumar, N. (2019). *Server-Less Rule-Based Chatbot Using Deep Neural Network (Dissertation)*.
- Shang, L., Lu, Z., & Li, H. (2015). Neural responding machine for short-text conversation. *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, 1, (pp. 1577-1586).
- Sherstinsky, A. (2020, March). Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network. “*Physica D: Nonlinear Phenomena*” : *Special Issue on Machine Learning and Dynamical Systems*, 404.
- Sojasingarayar, A. (2020). Seq2Seq AI Chatbot with Attention Mechanism.
- Song, Y., Li, C.-T., Nie, J.-Y., Zhang, M., Zhao, D., & Yan, R. (2018). An Ensemble of Retrieval-Based and Generation-Based Human-Computer Conversation Systems.
- Srivastava, N., Hinton, G., Krizhevsky, A., & Sutskever, I. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, 15, 1929-1958.
- Tao, C., Gao, S., Shang, M., & Wu, W. (2018). Get The Point of My Utterance! Learning Towards Effective Responses with Multi-Head Attention Mechanism. *Proceedings 27th International Joint Conference Artificial Intelligence*, 4418 - 4424.

- Trivedi, A., Gor, V., & Thakkar, Z. (2019). Chatbot generation and integration: A review. *International Journal of Advance Research, Ideas and Innovations in Technology*, 5(2).
- Vargas, R., Mosavi, A., & Ruiz, R. (2017). DEEP LEARNING: A REVIEW. *Advances in Intelligent Systems and Computing*, 5.
- Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A., . . . , & Polosukhin, I. (2017). Attention Is All You Need. *Advances in Neural Information Processing Systems*, 5998-6008.
- Vinyals, O., & Le, Q. (2015). A Neural Conversational Model. *International Conference on Machine Learning (ICML) Deep Learning Workshop*.
- Wu, Y., Li, Z., Wu, W., & Zhou, M. (2018). Response Selection with Topic Clues for Retrieval-based Chatbots. *Neurocomputing*, 316, 251-261.
- Zeiler, M. (2012). ADADELTA: An adaptive learning rate method.
- Zheng, A. (2015). *Evaluating Machine Learning Models* . O'Reilly.
- Zhou, H., Huang, M., Zhang, T., Zhu, X., & Liu, B. (2018). Emotional chatting machine: Emotional conversation generation with internal and external memory. *Proceedings 32nd Association for the Advancement of Artificial Intelligence Conference Artificial Intelligence*, 730-738.