SMU
SINGAPORE MANAGEMENT UNIVERSITY

**CS421 – Introduction to Machine Learning**

AY 2019/20 Term 2

# MeBot

*Using Deep Learning to Simulate Dialogue with Millennials*

## G1T1: Course Project Final Report

***Prepared for:***

*Professor Wang Zhaoxia*

*Instructor Tan Pang-Jin*

***Done by:***


Carlsten Lim **(01348212)**

Devyn Tan **(01318173)**

Keeve Quah **(01318313)**

Kwa Zhi Poh **(01334367)**

Lukas Tham **(01326617)**

## Table of Contents

## Abstract

This study examines the role of text messaging comprehension and the effects syntax has on NLP chatbots. Telegram data from our group members were taken and cleaned to fit using Seq2Seq. We introduced our project briefly with the motivations, problem statements as well as the Literature review found online.

We introduced how we perform Natural Language Processing. We explained that our team used at least 50k rows of questions and replies as the input data, with the pictures, links, and other mediums filtered out. Additionally, we also discussed what methodologies of Natural Language Processing we used to perform data pre-processing, such as tokenization, embedding process, to feed the data into the model. We also explained why we ignored the use of text stemming and lemmatization.

We discussed the different models used for our project. Through thorough planning, we took into consideration lots of models that had already been created, and utilised the commonly used algorithm for ChatBot models, as well as the newly created models to compare the performance between these models. We discuss the first Sequence To Sequence Recurrent Neural Network (RNN) models with Tensorflow and Keras library. We discuss the newly created Multi-head Attention Model. We discussed the Sequence To Sequence Recurrent Neural Network (RNN) models built with Tensorflow legacy code.

We discuss the results obtained from the three different models. Using the same dataset throughout the three models, we ran three different corpuses from different people in the group. Mainly it was Zhi Poh's, Lukas's and Keeve's dataset. Overall, we found that the Multi-head Attention Model performed the best out of the three models.

Finally, we discuss what we learnt from this project, the difficulty of tuning the hyperparameters for the different models, and the difficulty of implementing the code. From this project, we also explained the constraints we faced to create the ChatBot, and finally, how can we improve on our current ChatBot model in the future for industrial use.

# 1. Introduction

The main objective of the project is to create a personalized Chatbot that acts on a particular person's behalf to respond in a similar manner to incoming messages. There is growing interest in the use of chatbots both for personal and business. Studies have shown that the primary driving force for this increasing trend of automation in communication is better "productivity" (Brandtzaeg & Følstad, 2017).

## 1.1 Motivation

With chatbots on the rise (Nguyen, 2020), seeking new algorithms to improve machine learning accuracy is a key factor in ensuring that our limited resources are best utilised to maximise efficiency. One of the issues with most chatbots is that they lack a personal appeal (Guha, 2018). With that in mind, what we want to achieve is a chatbot that is capable and smart enough to simulate the behavioural answering patterns of a particular person. This will give the user a more personalised experience, as though he is talking to his friend.

## 1.2 Problem statement

Millennials lead busy lives, with multiple full-time commitments. The typical Millennial sends and receives about 128 texts daily, and 3853 texts monthly (Rufferty, 2017). Between university classes, part-time jobs, and extracurricular activities, more millennials are turning to chatbots to save precious minutes within a day. Creating a chatbot that texts like a Millennial would be invaluable to the entire generation.

However, given Millennial texting patterns that have many abbreviations and minimally accurate grammar syntax, using standard python packages and pre-trained data is simply not enough to accommodate the complexity of Millennial texting patterns. Hence, we decided to create a chatbot that could simulate a Millennial in the virtual space, using our own message history as our corpus.

# 2. Literature Review

To understand the algorithms used to build a chatbot, we began our research with condensed information from articles online. This allowed us to understand the main concepts behind Sequence to Sequence learning, as well as Attention, which we then implemented in our models.

To dive deeper into the specific hyperparameters to use for our models, we gained insight from specific research papers on chatbot neural networks, which will be discussed further in this section.

## 2.1 Overview of Chatbot Technology

Chatbots are "computer programs which conduct conversation through auditory or textual methods". This project aims to focus on human-machine interfacing on the textual front. Chatbot technology has seen major advancements in the past decade and has only recently incorporated deep learning as the primary approach in creating Chatbots. Currently, there are a huge number of chatbots created online. While many of these Chatbots are developed with very different algorithms, most are built on a handful of similar concepts in machine learning.

The Sequence to Sequence model (Seq2Seq) was first developed by Google to perform Machine Translation (Google, 2020). However, they are only used for re-ranking sentences instead of generating completely new ones. From there, many people start utilising Seq2Seq to create different models such as conversational models (Ghazvininejad et al. 2018)

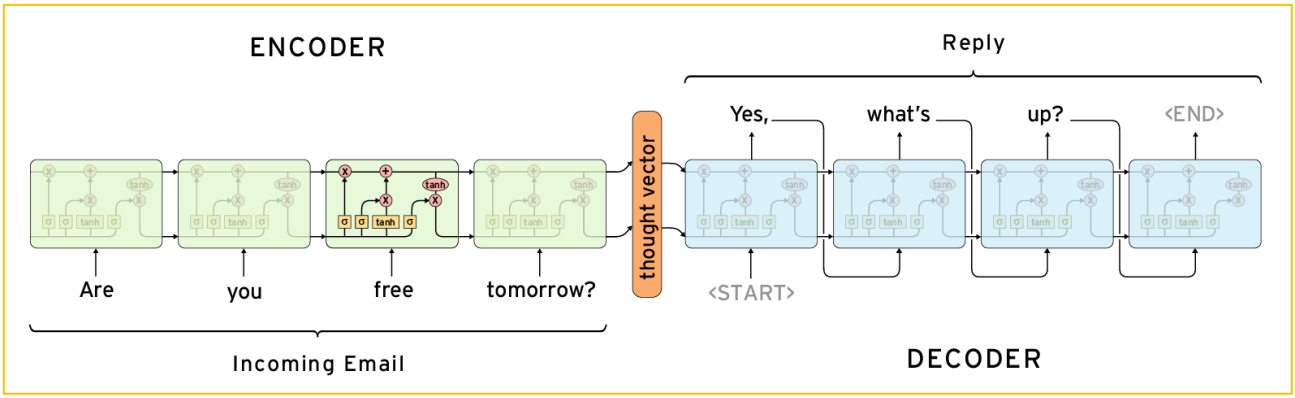


*Figure 1: RNN Architecture*

The Seq2Seq Model runs mainly on the Recurrent Neural Network (RNN). Seq2Seq is an encoder-decoder framework for Tensorflow and can be done by mapping an input sequence to an output sequence. The main idea is to make use of two RNN models that work together by predicting the next sequence given the previous sequence (Ma'amari, 2018).

Most deep learning methods use neural network architectures and Deep Neural Networks (DNN) can achieve state of the art accuracy. RNN is usually a feedforward neural network to sequences. It performs the same task for every element of a sequence, with assumption that the successive inputs are not independent of each other. RNN can usually pass signals between activation nodes in the same hidden layer. This allows RNN to store "memory" by passing the information gained during previous inputs to the current activation node. In our case, we are using a many to many process sequences which allows us to capture the sequence of words parse with the input gathered (Sutskever, Vinvyals, V. Le, 2018).

Given a sequence of input (x_1,x_2,x_3,…,x_n), RNN will calculate a sequence of outputs (y_1,y_2,y_3,…, y_n) based on the vectors:

$x_t$: inputs
$h_t$: hidden states (memory) ($s_t$ / h(t))
$o_t$: outputs
U: input weights →Wxh

V: hidden weights (recurrent connection) (W)→Wah/Whh
W: output weights (V)→Wao/Who/Why

For the Seq2Seq model, it will aim to map a fixed-length input with a fixed length output where the length of the input and output may be different. However, as mentioned before that RNN can store "memory" by passing the information gained during the process, it will tend to "forget" the first input due to resulting long term dependencies. Therefore, the Long Short-Term Memory (LSTM) is famous to remove long term dependencies. Therefore, LSTM will succeed in our Seq2Seq model (Wang, 2020).

$$p(y_1, \ldots, y_{T'} | x_1, \ldots, x_T) = \prod_{t=1}^{T'} p(y_t | v, y_1, \ldots, y_{t-1})$$

$Y_1, \ldots, y_{T'}$ = Output Sequence
$x_1, \ldots, x_T$ = Input Sequence
v = Vector Representation

The goal of LSTM is to estimate the conditional probability of the input sequence and its corresponding output sequence. It contains three gates, the input gate, output gate and the forget Gate. The input gate and the forget gate decide how the next internal state is influenced by the input as well as the last internal state respectively. Additionally, the forget gate tends to "Forget" inputs that are irrelevant. The forget gate is a neural network with the sigmoid function that calculates the probability of the input from 0 to 1 using the sigmoid function. If the probability input is more towards 0, it will tend to forget, while if its value is more towards 1, it will tend to remember it longer. On the other hand, the output gate uses a *tanh*

activation function, which determines how the output of the network is influenced by the internal state (Nguyen, 2018).

## 2.2 Specific Papers

***Sequence to Sequence Learning with Neural Networks - Sutskever, Vinyals and Le. Published in 2014.***

This paper discussed the basic architecture of an LSTM cell and compared the results of different LSTM architectures on machine translation tasks, using BLEU as an evaluation metric. It was found that an ensemble of five reversed LSTMs with beam size 12 obtained a BLEU score of 34.81, outperforming single reversed LSTM with beam size 12, which obtained a score of 26.17. An architecture of 5 LSTMs with beam size 1 obtained a score of 33.00.

Since machine translation using Seq2Seq learning and chatbots are both classification problem using argmax probability, we could use the best architecture reported in the paper for our own implementation.

As per section 3.4 of the paper, the best model implemented in the paper used deep LSTMs with 4 layers with 1000 LSTM units in each layer, with embedding dimensions of 1000 by 1000. The size of the dataset was a vocabulary of 160,000 (input) and 80,000 (output). Considering the significantly smaller vocabulary size of our data corpus at 8000 unique words, we decided to adjust these parameters to suit our purposes. We eventually settled on 3 LSTM layers with 112 LSTM units and embedding dimensions of 112 by 112 for our base model, the KeeveBot.

***Attention is All You Need - Vaswani, Shazeer, Parmar and others. Submitted in 2017.***

Unlike other pieces of literature, this paper used transformers based on feedforward neural networks with attention, as opposed to Seq2Seq learning, which virtually all commercial chatbots are developed on. We saw this architecture as an alternative to the two Seq2Seq models we would implement, and potentially outperform the other models.

| Model | BLEU | | Training Cost (FLOPs) | |
|---|---|---|---|---|
| | EN-DE | EN-FR | EN-DE | EN-FR |
| ByteNet [18] | 23.75 | | | |
| Deep-Att + PosUnk [39] | | 39.2 | | $1.0 \cdot 10^{20}$ |
| GNMT + RL [38] | 24.6 | 39.92 | $2.3 \cdot 10^{19}$ | $1.4 \cdot 10^{20}$ |
| ConvS2S [9] | 25.16 | 40.46 | $9.6 \cdot 10^{18}$ | $1.5 \cdot 10^{20}$ |
| MoE [32] | 26.03 | 40.56 | $2.0 \cdot 10^{19}$ | $1.2 \cdot 10^{20}$ |
| Deep-Att + PosUnk Ensemble [39] | | 40.4 | | $8.0 \cdot 10^{20}$ |
| GNMT + RL Ensemble [38] | 26.30 | 41.16 | $1.8 \cdot 10^{20}$ | $1.1 \cdot 10^{21}$ |
| ConvS2S Ensemble [9] | 26.36 | **41.29** | $7.7 \cdot 10^{19}$ | $1.2 \cdot 10^{21}$ |
| Transformer (base model) | 27.3 | 38.1 | **$3.3 \cdot 10^{18}$** | |
| Transformer (big) | **28.4** | **41.8** | $2.3 \cdot 10^{19}$ | |

*Figure 2: Evaluation of Architectures (BLEU and Training Cost)*

Like the previous paper, the model was applied on a translation task, with a German-to-English development dataset. It was noteworthy that the model achieved a reasonably good BLEU score at lower training costs than deeper neural networks (figure above). Given that our model would be implemented on Google Colab, training costs were very important to us, and we decided to implement the architecture outlined in section 5 of the paper. Further details of our implementation will be covered in a later section.

# 3. Datasets

## 3.1 Dataset used

The dataset used are Zhipoh's, Lukas's and Keeve's Telegram corpus. We trained our models based on each individual corpus. Zhipoh's corpus consists of 50K data, Keeve's corpus consists of 23K data and Lukas's corpus consists of 12K data. Using the NLTK library, Gensim Word2Vec, we worked closely with this library to convert the words to vector form.

## 3.2 Data Pre-processing

As the data structure of our data sets are determined by the application's design. Our first step was to take a wide overview of the datasets and analyse the necessary fields that we will use for our predictive model. The dataset was exported in JSON, machine readable format and for us to first analyse, we transformed the dataset into a Dataframe table and decapsulate the nested JSON stored as **message** as an extension to the Data Frame. This Dataframe is exported into a spreadsheet for human verification on the key columns of data.

The key columns of data identified for analysing are id, type, from and text as these data tells us who is the sender, inclusive of the user, of a message. For the scope of our project, we are primarily focusing on text processing and exclude all other media file such as images, videos and audio files.

We reduced the dimensions of the data excluding columns which headers are not listed in a list of headers names. This list of headers are essentially headers which want to preserve, columns headers that are not listed are headers of value which we will not be using as they are not within our focus of the project or have no values in the entire columns.

The next step is concatenating **text** data to distinguish between input and output to train the models while removing noise. The inputs will be words, phrases or sentences by other users while output will be words, phrases or sentences by the user. Therefore, we iterate through the Dataframe row by row, taking into account the state of whether the **text** data should be concatenating as the input field or output field with the consideration of noise removal to skip the current row of data.

**Text** data that includes encapsulated JSON tags would indicate that it contains hyperlinks or other objects such as contact card objects will not be concatenated. An empty **text** data would indicate it is a sticker and we will also exclude this data.

These inputs and outputs were initially stored in a dictionary as a key-value pair. However, we realised that this would replace inputs that are the same with the most recent response. Subsequently, we amended our methodology of storing the data in Dataframe and appending it directly to the main Dataframe instead of storing in a dictionary to improve computational time. The Dataframe will then be exported as a spreadsheet to allow different tweaks for different model training.

Essentially, we created a generally baseline training data consisting of input and output. At this stage, what is left of the data comprises words, punctuation and even emoji. Concatenated words capitalisation is also handled by changing the uppercase of the first character of the concatenating data, this done so that the response will look more realistic as a sentence since each text data has its first character to be uppercase when users sends out messages. For each model we train, we make use of the above generated baseline to suit each individual model.

# 4. Methodology and Models

We will be utilizing two different RNN with LSTM models with our telegram chat history to find out the most accurate Seq2Seq Chatbot model. We are utilizing all our telegram dataset which consist of normal text and emojis, with at least 50,000 data that will be parsed. Additionally, we will also use a Multi-head transformer Chatbot model as well to verify the accuracy of the RNN models.

## 4.1 Seq2Seq Model with Tensorflow Keras - LukasBot

The first model utilises more on the Tensorflow Keras libraries while the second model only utilizes Tensorflow libraries. Ultimately, we would like to find out which LSTM model produces a much more accurate and desirable output.

### 4.1.1 Encoder - LSTM

After performing our data pre-processing as well as data cleaning, the input and output data will then be parsed into the encoder of the RNN model using the Keras Tensorflow libraries.

The encoder is a stack of RNN with (LSTM) where it will each accept a single element of the input sequence, collect information for that element and perform forward propagation. The process of forward propagation is as such, the RNN will take in 2 inputs where x(t) is the input at time-step t, and a(t-1) is the hidden state

activation at previous time steps. They will both be multiplied together through matrix multiplication with different weight matrices. The results that are obtained will be then summed up with the bias to produce h(t). a(t), which is the hidden state activation at time step t, will be calculated by passing h(t) through an activation function that can be sigmoid, tanh or reLu. After which, a(t) will be then outputted by the RNN cell and given to the next RNN cell for computation (Wang, 2020).

Lastly, it will then compute o(t) with multiplying a(t) with Wao, the hidden-output weight matrix by adding together with the bias. Lastly, the yhat(t) is obtained, a vector of normalized probabilities over the true output o(t) by passing an activation function, if it's a categorical data, SoftMax loss function will be usually used (Wang, 2020)

*(1) h(t) = Wxh * x(t) + Wah * a(t-1) + bh*

*(2) a(t) = activation function (h(t))*

*(3) o(t) = Wao * a(t) + bo*

*(4) yhat(t) = activation function (o(t))*

#### 4.1.2 Dropout Layer

We also implemented a dropout layer. We found that the dropout layer is useful to reduce overfitting and improve the generalization of deep neural networks. A hyperparameter is introduced which specifies a probability which output layers are dropped out. For our case, we set that dropout rate to be 0.2. Dropout will identify all the layers of the network, retaining the units with the probability of higher than 0.2, those that are lower than 0.2 will be "dropped out" (Brownlee, 2019).

#### 4.1.3 Hyperparameters

*Batchsize = 300*

*Epochs = 350*

*Input dimensions = vocabulary size,*

*Output dimensions = 200*

*Dropout rate = 0.2*

*LSTM Layer = 200*

Some of the hyperparameters were adjusted to suit the needs for different types of dataset fed into the model. For example, Lukas's dataset has a vocabulary size of 3450, while Keeve's dataset has a vocabulary size of 2450, the hyperparameters will be set accordingly for our neural networks to take in the dataset. Based on the dataset. the hyperparameters are set to find the most optimum output.

#### 4.1.4 Training

The data collected were tokenized and broken down into individual vocabulary words. Due to the limitations of the amount of RAM given by Google Colab, we were only able to run the first 1000 rows of data for the LukasBot Tensorflow Keras LSTM Model as the embedding process done earlier was dependant on the vocabulary existing in the Word2Vec library. However, most of the words in the corpus are Singlish, and most of these Singlish words do not exist in the Word2Vec library. To rectify the embedding issue, we found that we cannot filter out words that do not exist in the library. This might cause the result to be inaccurate. Therefore, we came out with randomizing the individual corpus, this will allow the model to capture the individual Singlish word to match with the output.

With the 1000 rows of data, we decided to minimize the loss with our neural network model. However, we were also cautious not to overfit the data as well. Our team decided on how many epochs to run the model. An epoch is an iteration over the entire corpus parsed into the model. It consists of one full training cycle, this training cycle consists of performing both forward and backward propagation. Over time, the loss will decrease to create a more accurate output for the model. We also had to decide how many batch sizes are needed to train the model. The batch size refers to the number of samples that will be propagated through the network. To identify the ideal number of epochs as well as batch size, we utilised the concept of Gradient Descent Algorithm. The algorithm is one of the most popular algorithms to perform optimization, especially for neural networks. The algorithm uses derivatives in optimization problems. It decides whether to increase or decrease the weight. Based on the Cost Function,

$$J_{m,b} = \frac{1}{N}\sum_{i=1}^{N}(Error_i)^2$$

We will focus more on batch gradient descent as we are mostly working on the batch sizes to decide the optimal weights.

$$\frac{\partial}{\partial \mathbf{w}} u(\mathbf{w}, b, \mathbf{x}) = \begin{cases} \vec{0}^T & \mathbf{w} \cdot \mathbf{x} + b \leq 0 \\ \mathbf{x}^T & \mathbf{w} \cdot \mathbf{x} + b > 0 \end{cases}$$

We also have to consider the number of neurons in a single layer. We decided that the number of neurons in a layer should match the number of questions and answers.

To conclude, we found that the most optimal training for all the data is 350 epochs, 300 batch size.

We trained three different models for each person's data, for Zhi Poh's, Lukas's and Keeve's corpus. The in turn, will produce three different outputs to test its accuracy between different corpus. In total, the training for these three models took 3 days to complete as the model is only able to train each corpus at one time.

```
Epoch 338/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0289
Epoch 339/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0298
Epoch 340/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0284
Epoch 341/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0288
Epoch 342/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0283
Epoch 343/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0282
Epoch 344/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0282
Epoch 345/350
1454/1454 [==============================] - 30s 21ms/sample - loss: 0.0279
```

*Figure 3: Screenshot of Training Process*

### 4.1.5 Loss and Optimizer

Once the forward propagation process is complete, it will perform the backward propagation to improve the accuracy of the model, the backward propagation begins by calculating the loss made by yhat(t), it will perform the reverse method of forward propagation, however, using the derivatives of the calculated weights so as to minimize the loss. Within the RNN model, our model will also contain the LSTM model, which will help to decide whether to retain the input value, or to "forget" the input value. This will help to create a much accurate model as without the LSTM function, RNNs tend to have long term dependencies. The LSTM will concatenate the hidden state value as well as the current input. The concatenated value will then be fed into the "Forget" gate. This layer will remove irrelevant data. A candidate layer is created to pass through the concatenated value, this candidate layer will hold all possible values to add into the cell state. Afterwhich, this value will be fed into the input layer and decide whether this value should be added into the new cell state. After passing through the three layers, the forget gate, candidate layer and input

layer, it will be calculated through matrix multiplication with the previous cell state. The output will then be computed through the output layer. These processes will be performed similarly for both for the encoder and decoder.

#### 4.1.6 Limitations of the LSTM Model

Based on the theoretical studies found online, we implemented the model as per explained above. However, there are some flaws that we discovered while building the model. At first, we believe that LSTM is capable of solving problems with long term dependencies. However, due to the limitations of the amount of RAM given by Google Colab, further adding that we have to randomize the tokenization for each character, we have a lot of parameters to train for our models. Therefore, we had to cut down the amount of data used to run the model to about a 1,000 dataset. We found that LSTM works better with a big dataset. Therefore, the accuracy of the second model of Seq2Seq might be more accurate than the first model due to the amount of data that can be fed into the model for training.

### 4.2 Seq2Seq Model with Tensorflow - KeeveBot

#### 4.2.1 Encoder

After performing our data pre-processing as well as data cleaning, the input and output data will then be parsed into the encoder of the RNN model, which in this case is an LSTM encoder with TensorFlow's *nn.rnn_cell.BasicLSTMCell() object.*

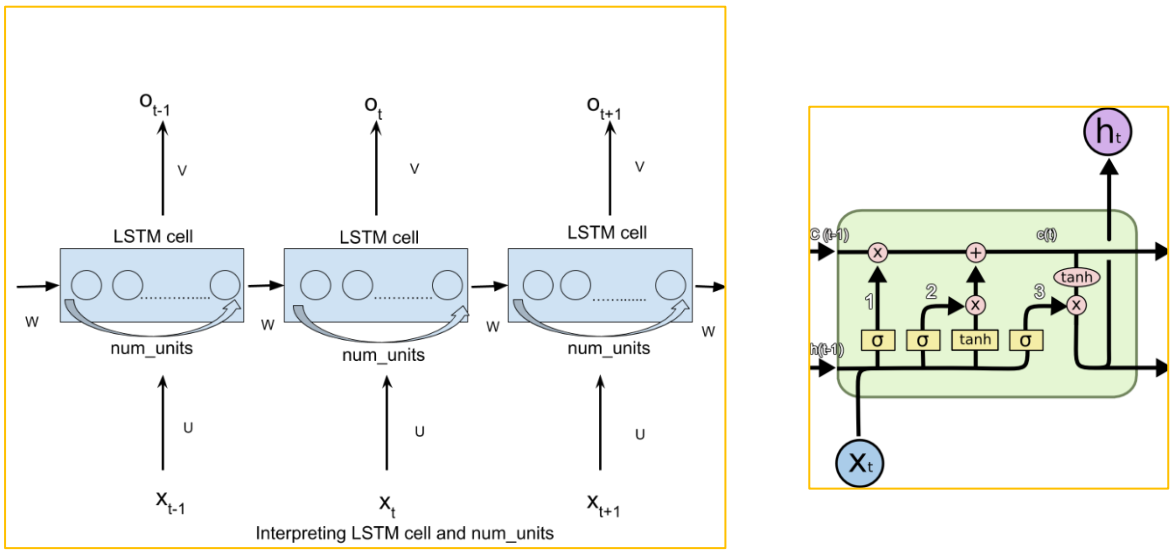


*Figure 4: LSTM Architecture*

This LSTM encoder, according to the Tensorflow documentation of the function, encodes each word with using an LSTM cell as explained in the previous model. The diagram above illustrates what happens to each

word as it passes through an LSTM cell. The circles within the LSTM cell are LSTM units, which are activated using the function illustrated in the diagram on the right.

The key input parameter of the function is the number of LSTM units in the cell, which was a hyperparameter that we had to choose carefully. Too many units and the model would overfit; too little and the model would underfit. Considering the recommendations from several academic articles and the documentation, we chose 112 as our number of LSTM units given our limited training time of just 12 hours using Google Colab.

### 4.2.2 Hyperparameters

*BatchSize = 24*

*maxEncoderLength = 15*

*maxDecoderLength = maxEncoderLength*

*lstmUnits = 112*

*embeddingDim = lstmUnits*

*numLayersLSTM = 3*

*numIterations = 300000*

We tweaked different hyperparameters to fit our resource constraints. Unfortunately, we were limited greatly by Google Colab's GPU constraints. For instance, a batch size of 24 (24 input and output sentences) was the highest we could choose without crashing our code.

We chose our number of LSTM units, which in this case is the variable *numLayersLSTM* above, to be 3, as was the recommended number outlined in the documentation for Tensorflow's *embedding_rnn_seq2seq()* function. This variable is used later in the model to encode inputs and decode predictions. Essentially, this model uses the same LSTM structure in both encoding and decoding.

### 4.2.3 Training

The model is trained on the number of iterations, instead of the concept of epochs, which is used in most machine learning models. This was done to accommodate our use of the *embedding_rnn_ seq2seq()* function, which was intended to be the crux of this model. In each iteration, the function selects a batch of 24 X and Y training sentences from Seq2SeqXTrain.npy and Seq2SeqYTrain.npy respectively.

Forward and backward propagation are run on these inputs as similar to the previous model, and loss is calculated every 50 iterations. Given the flexibility of running our model on iterations, we could feed in test inputs during iterations.

Test Inputs - ["keeve", "hi","hey how are you", "that girl quite chio sia","how do this question"]. Our test inputs were deliberately chosen to incorporate Singaporean millennial slang to see how our model was learning from the data corpus, and if it started to sound anything like us.

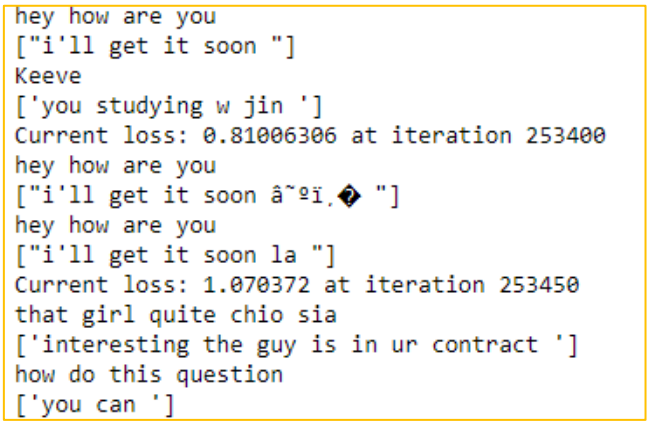


*Figure 5: Screenshot of Training Process (KeeveBot)*

Each string in the array could be encoded and decoded to show the predictions of the model while it was being trained. Above is a sample of what the model predicted at iterations > 250,000. Additional screenshots of the predictions to the input strings at different iterations are attached in the Appendix.

### 4.2.4 Loss and Optimizer

We calculated the loss using TensorFlow's contrib.legacy_seq2seq.sequence_loss() function, which - despite being legacy code - is still widely used in the Seq2Seq chatbots today. The optimizer we used in this model is TensorFlow's train.AdamOptimizer(1e-4).minimize(loss) function.

### 4.3 Multi-Layer Attention Model - ZhipohBot

Instead of recurrent neural networks, we explored an alternative which is referenced in the research paper "Attention is all you need" (Vaswani et al., 2017). The attention architecture can be built using neural networks similar to Seq2Seq models; it also has Encoding and Decoding layers to decipher the text data. The difference is that instead of using recurrent neural networks or convolutional neural networks, multi-head attention layers which consist of multiple scaled dot-product attention are used. This allows for parallelization by substituting the recurrence with attention and adding the text position in the form of positional encoding thereby accelerating the training.

$$Attention(Q, K, V) = softmax_k(\frac{QK^T}{\sqrt{d_k}})V$$

Attention focuses on capturing the context and relevance of each word to the input queries through mass matrices multiplication.

Where Q or query is the vector representing the input word, and K and V are the vectors representing the input sequence, do note however that the K and V vectors may represent different meanings in both the encoder and decoder.

On the encoder portion, we first do a dot product of the query(Q) word matrix to a transpose of the key(K) input sequence matrix. The dot-product attention is scaled by a factor of square root of the depth to regularize the dimensionality before we use SoftMax as an loss function. Finally, a dot product again to the value(V) input sequence matrix to get a weighted sum of values with the context and relevance of the input query on the input sequence.

In order to speed things up, multi-head attention refers to stacking multiple heads (each representing features of a word) in a stack to generate a matrix with multiple attention heads that captures the different features in the message. This allows the model to run in parallel instead of a sequence, improving the speed the model can be trained in.

$$\text{head} = \text{Attention}(QW^Q, KW^K, VW^V)$$

The heads are then concatenated and transformed using an output weight matrix.

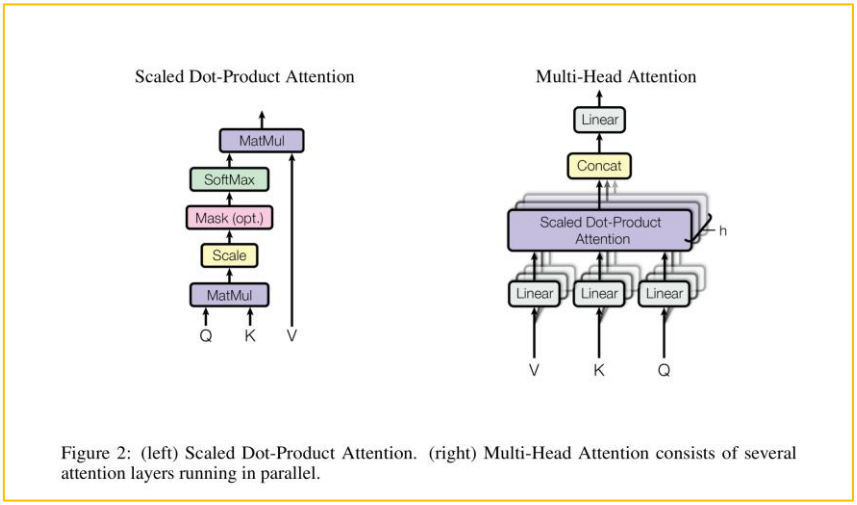


Figure 2: (left) Scaled Dot-Product Attention. (right) Multi-Head Attention consists of several attention layers running in parallel.

*Figure 6: Scaled Dot-Product Attention (left), Multi-Head Attention (right)*

$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

From here we build an attention neural network architecture based on multi-head attention on google colab.

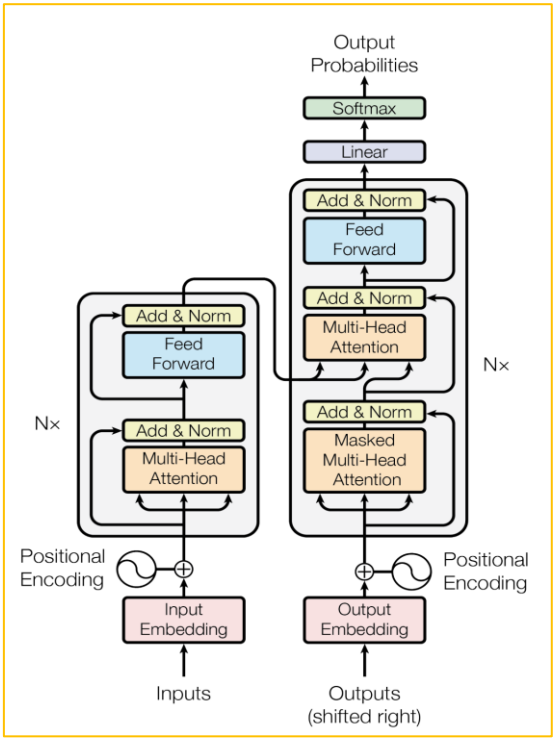


*Figure 7: Feedfoward NN with Multi-head Attention*

On another level of abstraction, we can then build our Attention-based neural network with the architecture consisting of both an encoder and decoder for our text-messaging chatbot.

### 4.3.1 Encoder

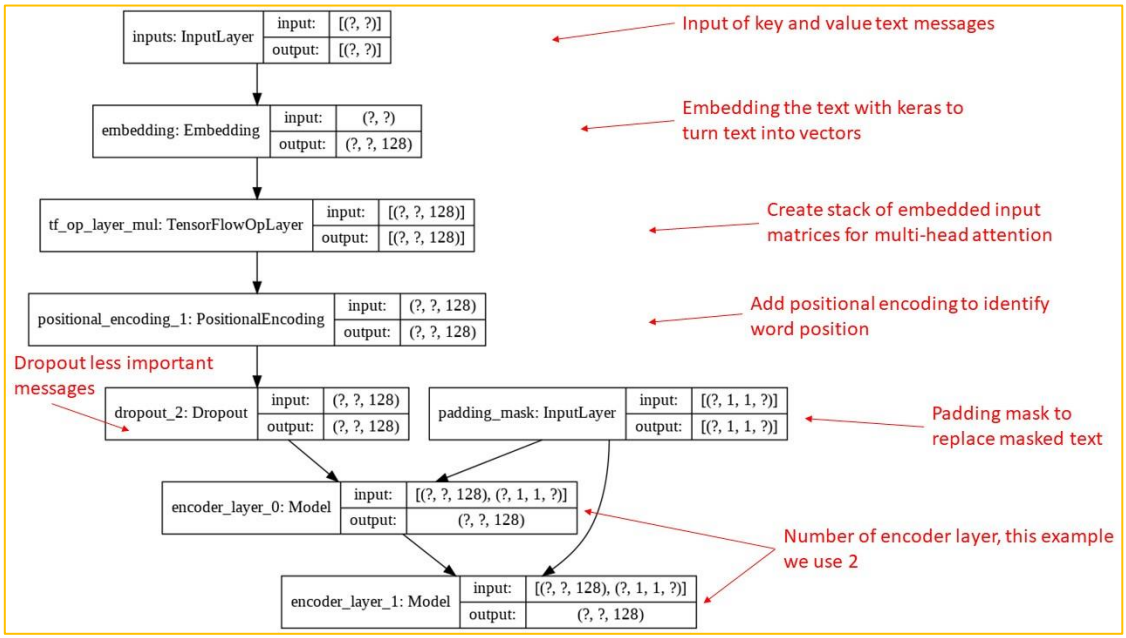


*Figure 8: Encoder (ZpBot)*

Encoders help to convert the text messages into matrices which we can compute using attention. Our input layer consists of the text message data which has been previously processed and cleaned into an input and output. We used a Keras embedding layer to convert the text data into a vector and added this with another positional encoding layer which captures the positional dependencies of the words data relative to each other, replacing the sequential nature of recurrent neural networks. The model then goes to the attention process and feedforward the information about the context and features of each word input into the decoder.

### 4.3.2 Decoder

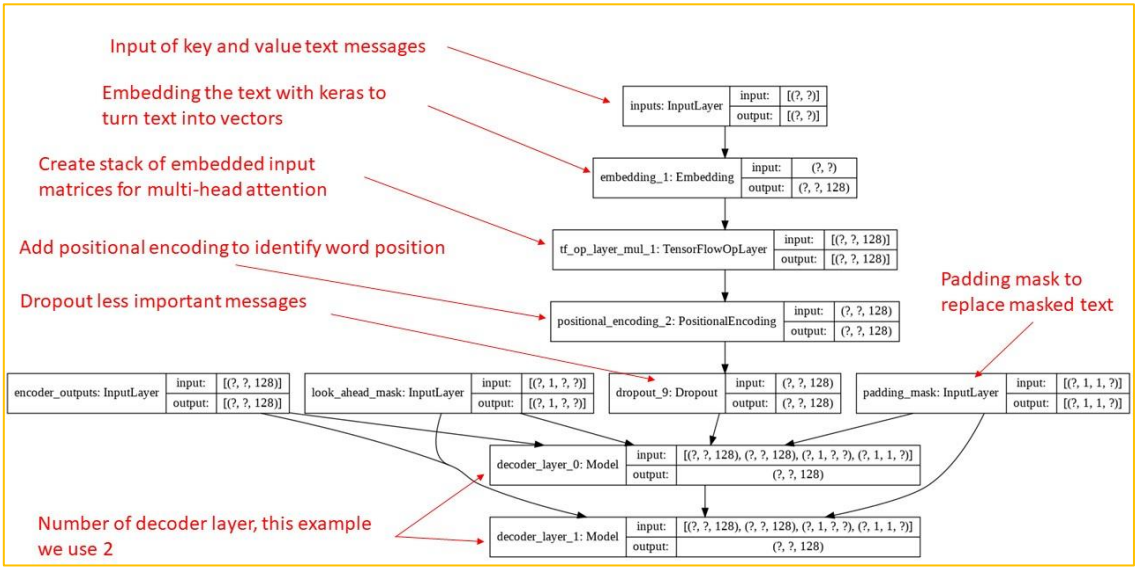


*Figure 9: Decoder (ZpBot)*

Decoders works mostly with the output text messages matrices, we used a masking layer which helps to improve the prediction of correct outputs when masked. The information passed from the encoder is then calculated in relation to the attention output matrices. Lastly, we flatten this information and use SoftMax loss function to obtain the optimal outputs and weights.

### 4.3.3 Hyperparameters

*Vocabulary size = ~ 8000+*
*Num_layers = 2*
*D_model = 256*
*Num_heads = 8*
*Units = 512*
*Dropout = 0.1*
*Batch = 64*
*Buffer Size = 20000*

We tweaked different hyperparameters to fit our resource constraints. Unfortunately, we were limited greatly by Google Colab's GPU constraints. For instance, a batch size of 64 (64 input and output sentences) was the highest we could choose without crashing our code.

### 4.3.4 Loss and Optimizer

We calculated the loss using Sparse Categorical Cross Entropy because we consider each class to be mutually exclusive, logarithmic loss because we are doing classification of our text messages. The optimizer we used in this model is TensorFlow's train.AdamOptimizer(1e-4).minimize(loss) function.

## 4.4 Differences in Seq2Seq Models KeeveBot and LukasBot

| | LukasBot | KeeveBot |
|---|---|---|
| Size of Dataset | 1000 rows<br>2000 vocabulary size | 23,894 rows<br>38,950 vocabulary size |
| Preprocessing | result.json: Telegram chat history in JSON format. Based on Lukas' telegram data.<br><br>Text_to_sequence: vectorize a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf<br><br>Utils_to_category: Converts a class vector (integers) to binary class matrix.<br><br>Encoder: One hot encoding of the input word and measure the output error compared to one hot encoding of the target word,<br><br>Padding: This function transforms a list of num_samples sequences (lists of integers) into a 2D Numpy array of shape (num_samples, num_timesteps). num_timesteps is either the maxlen argument if provided, or the length of the longest sequence otherwise. | result.json: Telegram chat history in JSON format. Based on Keeve's telegram data.<br><br>preprocessing.ipynb: converts result.json into a dictionary with key value pairs:<br>- Key: Keeve's responses to a message<br>- Value: a message sent by another person<br>- Sample key-value pair: {*'yea man good luck for db bro': 'all the best bro hahaha u ready'*}<br><br>The dictionary is saved as response_dict.npy. A text file, response_dict.txt, is created from the dictionary. The file essentially saves all the items (key and values) as a single long text string so that a bag of words can be derived in a later method. |
| Embedding | Embedded with Tensorflow Keras Seq2Seq | Embedded with TensorFlow's *embedding_rnn_seq2seq ()* function. |
| Models | Tensorflow Keras model<br><br>One word, one neuron.<br><br>Dropout rate: 0.2 | Encoder: TensorFlow's *nn.rnn_cell.BasicLSTMCell()* object.<br><br>Loss is calculated using TensorFlow's *contrib.legacy_seq2seq.sequence_loss()* function.<br><br>Optimizer is TensorFlow's *train.AdamOptimizer(1e-4).minimize(loss)* function. |

| | | |
|---|---|---|
| Hyperparameters | *Batchsize = 300*<br>*Epochs = 350*<br>*Input dimensions = vocabulary size,*<br>*Output dimensions = 200*<br>*Dropout rate = 0.2*<br>*LSTM Layer = 200* | *BatchSize = 24*<br>*maxEncoderLength = 15*<br>*maxDecoderLength = maxEncoderLength*<br>*lstmUnits = 112*<br>*embeddingDim = lstmUnits*<br>*numLayersLSTM = 3*<br>*numIterations = 300000* |

## 5. Results

The team rated the chatbot based on how similar it speaks like the person it is supposed to as well as whether the reply makes sense. The scale of the score is as such: 1 being that the reply makes no sense and is totally different to how that person would reply and 8 being that the reply makes full sense and it is exactly how that person would reply. The Likert scale of 1 to 8 is used here as there is no absolute way to measure the accuracy of a chatbot. As for the range, having a small range would lose some meaning if the reply is very good instead of just good or excellent. Having a large range would create too many variances to get an accurate picture as 14 to one person may be a 15 to another.

### 5.1 LukasBot Results

| Data Corpus Used | Score |
|---|---|
| *Keeve's Telegram Data* | First Question: 4<br>Second Question: 4<br>Third Question: 7<br>Fourth Question: 5<br>Fifth Question: 7<br>Sixth Question: 3<br>Seventh Question: 5<br>Eighth Question: 7<br>Ninth Question: 8<br>Tenth Question: 6<br>**Total: 56** |

```
Enter question : Sup bro
 nah i knocked out end
Enter question : Wanna grab lunch?
 nah i knocked out end
Enter question : How was the lesson
 i just know you went and puked sometime end
Enter question : Bro this qns how to do?
 nah man end
Enter question : LMAO ok
 you coming rite end
Enter question : Have you recorded down what we are suppose to do?
 nah i knocked out end
Enter question : Can you send me the code
 nah i figured since you toh alrd i probably shouldn t drink more haha end
Enter question : 😀 😁 😂 🤣 😃
 you got it hahaha i guess end
Enter question : Do you think it will work out?
 i just took a part of our twc report and made it sound like shit haha end
Enter question : Damn weird eh
 you get all the luck you mofo end
```

| Data Corpus Used | Score |
|---|---|

*Zhi Poh's Telegram Data*

```
Enter question : sup bro
 i think i did the prof work tgt and all the project down end
Enter question : wanna grab lunch?
 i think i did the prof work tgt and all the project down end
Enter question : how was the lesson?
 i think i did the prof work tgt and all the project down end
Enter question : bro this qn how to do?
 i think i did the prof work tgt and all the project down end
Enter question : LMAO ok
 i think i rejected cos i only transferred in pdf okie end
Enter question : have you recorded down what you suppose to do?
 i think i did the prof work tgt and all the project down end
Enter question : can you send me the code
 i think i did the prof work tgt and all the project down end
Enter question : 😀😃😄😁
 i think i did the prof work tgt and all the project down end
Enter question : Do you think it will work out?
 i think i did the prof work on you to chiong time end
Enter question : damn weird eh
 i think i did the prof work tgt and all the project down end
```

First Question: 1

Second Question: 2

Third Question: 1

Fourth Question: 1

Fifth Question: 1

Sixth Question: 4

Seventh Question: 1

Eighth Question: 1

Ninth Question: 1

Tenth Question: 1

**Total: 14**

*Lukas' Telegram Data*

```
Enter question : Sup bro
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
Enter question : Wanna grab lunch?
 i think i know where end
Enter question : How was the lesson?
 i think i know where end
Enter question : Bro this qns how to do?
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
Enter question : LMAO ok
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
Enter question : Have you recorded down what we are supposed to do?
 i think dont have leh end
Enter question : Can you send me the code
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
Enter question : 😀😁😄🤣😃
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
Enter question : Do you think it will work out?
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
Enter question : Damn weird eh
 i think we do together very luan you ctrl f isit haha nvm i dont touch first i go link the websites end
```

First Question: 1

Second Question: 2

Third Question: 1

Fourth Question: 1

Fifth Question: 1

Sixth Question: 4

Seventh Question: 1

Eighth Question: 1

Ninth Question: 1

Tenth Question: 1

**Total: 14**

## 5.2 KeeveBot Results

| Data Corpus Used | Score |
|---|---|

*Lukas' Telegram Data*

```
Sup bro
can what u have microsoft rm on friday thanks

Wanna grab lunch
no no ct ma ct ma

How was the lesson
wah

Bro this qns how to do
okie show him

LMAO ok
no midterms i want to be a retard the is on next week haha yep

Have you recorded down what are we suppose to do
ofc quok to go man oh u doing the notifications now

Can you send me the code
okayy then okayy okayy okayy countable ya

😄😊😉😍😘
u got slot sch

Do you think it will work out
u do properly

Damn weird eh
pangseh kia
```

First Question: 1

Second Question: 1

Third Question: 3

Fourth Question: 2

Fifth Question: 1

Sixth Question: 6

Seventh Question: 3

Eighth Question: 1

Ninth Question: 8

Tenth Question: 3

**Total: 29**

*Keeve's Telegram Data*

```
Sup bro
eh what's your class class at the fountain can

Wanna grab lunch
where u

How was the lesson
yo you need to study for the construction

Bro this qns how to do
ok bro i'm not going to get a drink then hahaha

LMAO ok
eh sim is the platform i seeing u later

Have you recorded down what are we suppose to do
oh ok

Can you send me the code
eh pls i got some questions you know

😄😊😉😍😘
yo you know reubs

Do you think it will work out
okok thanks

Damn weird eh
or not 1pm
```

First Question: 2

Second Question: 8

Third Question: 3

Fourth Question: 5

Fifth Question: 1

Sixth Question: 5

Seventh Question: 8

Eighth Question: 5

Ninth Question: 3

Tenth Question: 3

**Total: 43**

*Zhi Poh's Telegram Data*

```
Sup bro
thanks bro

Wanna grab lunch
okie i go find u

How was the lesson
more speed

Bro this qns how to do
lol no prob i think sia

LMAO ok
lol

Have you recorded down what are we suppose to do
for wednesday 2 is at 2 station

Can you send me the code
where u know wat

😄😊😉😍😘
okie shoes like my class end

Do you think it will work out
thanks for ur day

Damn weird eh
okie
```

First Question: 4

Second Question: 8

Third Question: 3

Fourth Question: 3

Fifth Question: 7

Sixth Question: 2

Seventh Question: 1

Eighth Question: 1

Ninth Question: 2

Tenth Question: 6

Total: 37

## 5.3 ZPBot Results

| Data Corpus Used | Score |
|---|---|

*Keeve's Telegram Data*

```
Sup bro
 take 10 min bid was a nap then set up oh btw i bid at 10

Wanna grab lunch?
 sure meet you at pantry

How was the lesson
 ur type ah

Bro this qns how to do?
 a not yet i wanted to update you on both on one but i got a consult now

LMAO ok
 the entire section is copy paste

Have you recorded down what we are supposed to do?
 my ppl whew dk if I have a guy in my grp pass A for money but hAHA

Can you send me the code
 send me the link pls

😭 😆😁😉😁
 i got A fmi A- corp fin damn sian unless it's A+ fmi and A- corp fin that's the only 2 possibilities

Do you think it will work out?
 i'm not ah i mean it's ok man but yea I'll be at labs k

Damn weird eh
 lOL bro
```

Frist Question: 1

Second Question: 8

Third Question: 3

Fourth Question: 8

Fifth Question: 7

Sixth Question: 1

Seventh Question: 7

Eighth Question: 8

Ninth Question: 8

Tenth Question: 7

**Total: 58**

| Data Corpus Used | Score |
|---|---|

*Zhi Poh's Telegram Data*

```
Sup bro
 sri Lanka?

Wanna grab lunch?
 i at new free mrt lol end can

How was the lesson
 but after more than half the lessons the

Bro this qns how to do?
 friday is first order of 4

LMAO ok
 knn clip nails then fk

Have you recorded down what we are supposed to do?
 cliffen, house oso at IS student life to drink ah

Can you send me the code
 no nid say thanks hahahahaha ur school sent the right entrance cos it wasnt able to find experience ur gal done as well okay wif work

😭 😆😜🙂😁
 no more no more

Do you think it will work out?
 then not sure if I dun do too little date will start startup and block 2 directly

Damn weird eh
 s3.40 or 6.30pm idm give u the rest of us
```

First Question: 1

Second Question: 6

Third Question: 4

Fourth Question: 2

Fifth Question: 2

Sixth Question: 1

Seventh Question:2

Eight Question: 6

Ninth Question: 5

Tenth Question: 3

**Total: 32**

*Lukas' Telegram Data*

```
Sup bro
 so far no la i talk to u about the sch ah like this

Wanna grab lunch?
 where? i need singapore poly study later

How was the lesson
 cock

Bro this qns how to do?
 ya i got alot of ppl realise holiday hAHAH

LMAO ok
 ok la crazy la some ppl will talk to him by more impt  that that alright

Have you recorded down what we are supposed to do?
 eh piao

Can you send me the code
 ok wait ah i meet u at 230 can tmr?

😭 😆😜😉😁
 cause i saw

Do you think it will work out?
 lOL

Damn weird eh
 yea the bug metrics u updated right?
```

First Question: 1

Second Question: 8

Third Question: 8

Fourth Question: 4

Fifth Question: 5

Sixth Question: 2

Seventh Question: 8

Eighth Question: 2

Ninth Question: 7

Tenth Question: 3

**Total: 46**

The table below shows the average score for all three models:

| LukasBot | KeeveBot | ZPBot |
|---|---|---|
| 28/80 | 36/80 | 45/80 |

### 5.4 Discussion of Results

ZPBot managed to get the highest score of the three. Further test shows that it:

1. Has Personalisation

```
Input: hello
Output: konnichiwa
```

2. Can Understand Context & Longer Query

```
Input: Wanna go grab a drink?
Output: sure. 9am at cityhall?
```

3. Can Interpret Singlish

```
Input: walao eh no go sia
Output: sorry see u mah hahaha hope u even more busy day for my dates
```

The team suspects that certain limitations may have resulted in the multi head attention model (ZPBot) to outperform the other 2 models. First, it is the limited ram size of 25GB on Google Colab. This restricts the training vocabulary size greatly, especially so when LukasBot uses one hot encoding. Also, the server forces timeout every 12 hours which limits the training time of the Sequence-to-Sequence Models (LukasBot and KeeveBot) to that number of hours.

As for reasons of not using BERT as originally planned, the team does not have sufficient personal data to retrain the BERT model. Also, it is very expensive to do so both computationally and price wise. These 2 fundamental reasons strongly deterred the team from using BERT. The reason for not refining the current BERT model is because of the primary use case of this project, which is personalisation. Fine-tuning the current BERT will result in the loss of this very aspect. The ability to pick up Singlish and to speak like the person itself will not come true. It will, in fact, reply more like news articles.

# 6. Conclusion and Future Works

## 6.1 Learnings

Through this experience, the team had the opportunity to experience the difficulty in both understanding and coding the deep learning algorithms. The importance and knowledge of hyper-parameter was appreciated. Finally, the team learnt to better manage their expectations. With the limited computational power, one has to be realistic in setting the goals for a 6-week project.

## 6.2 Future Works

The team hopes to expand ZPBot by making it a Bi-directional Multi Head Attention model. Also, the team would like to be able to deploy the chatbot onto a platform such as telegram. Opportunities to commercialise this chatbot for large-scale business use can also be explored. This will bring forth a personal touch to consumers. Lastly, the team wishes that this chatbot can be used in place of a family member or loved one that has passed on. The wisdom and legacy of this loved one will then be able to last forever.

## 6.3 Conclusion and Acknowledgements

The team is satisfied with the results of each of our models given this limited time of the project. Though not perfect, ZPBot is able to mimic the texting of a person when trained on his dataset at least 50% of the time confidently. It was truly an eye-opening experience to the world of machine learning. To conclude, the team would like to thank Professor Wang Zhaoxia and Instructor Tan Pang Jin for the advice, guidance and support in this project as well as the module. Without their quick replies and help, this achievement would not have been possible.

# References


Agarwal, R. (2019, December 25). Chatbots aren't as difficult to make as You Think. Retrieved from https://towardsdatascience.com/chatbots-arent-as-difficult-to-make-as-you-think-f7f90255b993

Brandtzaeg P.B., Følstad A. (2017) Why People Use Chatbots. In: Kompatsiaris I. et al. (eds) Internet Science. INSCI 2017. Lecture Notes in Computer Science, vol 10673. Springer, Cham

Brownlee, J. (2019, August 6). A Gentle Introduction to Dropout for Regularizing Deep Neural Networks. Retrieved from https://machinelearningmastery.com/dropout-for-regularizing-deep-neural-networks/

Chauhan, N. S. (2019, May 2). Build your first chatbot using Python NLTK. Retrieved from https://towardsdatascience.com/build-your-first-chatbot-using-python-nltk-5d07b027e727

Chauhan, N. S. (2020, March 1). Let's build an Intelligent chatbot. Retrieved from https://towardsdatascience.com/lets-build-an-intelligent-chatbot-7ea7f215ada6

D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv*:1409.0473, 2014

Davis, T. (2020, January 24). How Chatbots Can Be Used to Improve Any Business. Retrieved from https://towardsdatascience.com/how-chatbots-can-be-used-to-improve-any-business-f7f3a1ebcbba

Deshpande, A. (2018). How I Used Deep Learning To Train A Chatbot To Talk Like Me (Sorta). Retrieved from https://adeshpande3.github.io/How-I-Used-Deep-Learning-to-Train-a-Chatbot-to-Talk-Like-Me

Ghazvininejad. M, Brockett. C, Chang. M-W, Dolan. B, Gao. JF, Yih. W-T, Galley. M. (2015, November 15). A Knowledge-Grounded Neural Conversation Model. Retrieved from https://arxiv.org/pdf/1702.01932.pdf

Google. (2020). Seq2Seq Model. Retrieved from https://google.github.io/seq2seq/

Gu, M. (2018, October 1). Building a Simple Chat Bot using Deep Learning. Retrieved from https://towardsdatascience.com/building-a-simple-chat-bot-using-deep-learning-6891a92ce87e

H. Schwenk. University le mans. http://www-lium.univ-lemans.fr/schwenk/cslm_joint_paper/, 2014.

Karani, D. (2018, September 2). Introduction to Word Embedding and Word2Vec. Retrieved from https://towardsdatascience.com/introduction-to-word-embedding-and-word2vec-652d0c2060fa

Krishnan, S. (2018, December 11). Chatbots are cool! A framework using Python. Retrieved from https://towardsdatascience.com/chatbots-are-cool-a-framework-using-python-part-1-overview-7c69af7a7439

Line-of-Code Completions, Cloudless Processing. (Dec 2014). Retrieved from https://kite.com/python/docs/tensorflow.contrib.legacy_seq2seq.embedding_rnn_seq2seq

Ma'amari, M. (2018, November 16). NLP: Sequence to Sequence Networks: Part 2: Seq2seq Model (EncoderDecoder Model). Retrieved from https://towardsdatascience.com/nlp-sequence-to-sequence-networks-part-2-seq2seq-model-encoderdecoder-model-6c22e29fd7e1

Nguyen, M. (2019, July 10). Illustrated Guide to LSTM's and GRU's: A step by step explanation. Retrieved from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

Q.V. Le, M.A. Ranzato, R. Monga, M. Devin, K. Chen, G.S. Corrado, J. Dean, and A.Y. Ng. Building high-level features using large scale unsupervised learning. In ICML, 2012.

Saxena, N. (2020, February 1). Chatbot Tutorial: Choosing the Right Chatbot Architecture. Retrieved from https://towardsdatascience.com/chatbot-tutorial-choosing-the-right-chatbot-architecture-5539c8489def

Sequence to Sequence Learning with Neural Networks. (2014). ArXiv. Retrieved from https://arxiv.org/pdf/1409.3215.pdf

Sidhu, R. (2019, October 17). How to Build a Chatbot - A Lesson in NLP. Retrieved from https://towardsdatascience.com/how-to-build-a-chatbot-a-lesson-in-nlp-d0df588afa4b

Sutskever, I, Vinvyals, O, V. Le Q, (2018). Sequence to Sequence Learning with Neural Networks Retrieved from https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. Advances in Neural Information Processing Systems, 2017-Decem(Nips), 5999–6009. ArXiv. Retrieved from https://arxiv.org/pdf/1706.03762.pdf

Wang, ZX, (2020). CS421 Introduction to Machine Learning [PDF File] Retrieved from https://elearn.smu.edu.sg/d2l/le/content/256803/Home

Xu, J. (2020, January 11). How To Create A Chatbot with Python & Deep Learning In Less Than An Hour. Retrieved from https://towardsdatascience.com/how-to-create-a-chatbot-with-python-deep-learning-in-less-than-an-hour-56a063bdfc44