A Comparative Analysis of Generative Neural Attention-based Service Chatbot

Sinarwati Mohamad Suhaili[1], Naomie Salim[2], Mohamad Nazim Jambli[3]
Pre-University, Kota Samarahan, Sarawak, Malaysia[1]
Faculty of Computing, Universiti Teknologi Malaysia, 81310, Skudai, Johor, Malaysia[1,2]
UTM Big Data Centre, Ibnu Sina Institute for Scientific and Industrial Research,
Universiti Teknologi Malaysia, 81310, Skudai, Johor, Malaysia[2]
Faculty of Computer Science and Information Technology, Universiti Malaysia Sarawak,
Kota Samarahan, Sarawak, Malaysia[3]

***Abstract*—Companies constantly rely on customer support to deliver pre-and post-sale services to their clients through websites, mobile devices or social media platforms such as Twitter. In assisting customers, companies employ virtual service agents (chatbots) to provide support via communication devices. The primary focus is to automate the generation of conversational chat between a computer and a human by constructing virtual service agents that can predict appropriate and automatic responses to customers' queries. This paper aims to present and implement a seq2seq-based learning task model based on encoder-decoder architectural solutions by training generative chatbots on customer support Twitter datasets. The model is based on deep Recurrent Neural Networks (RNNs) structures which are uni-directional and bi-directional encoder types of Long Short-Term Memory (LSTM) and Gated Recurrent Units (GRU). The RNNs are augmented with an attention layer to focus on important information between input and output sequences. Word level embedding such as Word2Vec, GloVe, and FastText are employed as input to the model. Incorporating the base architecture, a comparative analysis is applied where baseline models are compared with and without the use of attention as well as different types of input embedding for each experiment. Bilingual Evaluation Understudy (BLEU) was employed to evaluate the model's performance. Results revealed that while biLSTM performs better with Glove, biGRU operates better with FastText. Thus, the finding significantly indicated that the attention-based, bi-directional RNNs (LSTM or GRU) model significantly outperformed baseline approaches in their BLEU score as a promising use in future works.**

***Keywords*—Sequence-to-sequence; encoder-decoder; service chatbot; attention-based encoder-decoder; Recurrent Neural Network (RNN); Long Short-Term Memory (LSTM); Gated Recurrent Unit (GRU); word embedding**

I. Introduction

Providing excellent customer service while engaging with their clients has become more pivotal than ever in today's digitally connected era. Companies engage with customers to assist them with pre-post sale items regularly upgraded due to technological advancements or the communication revolution. Over the years, face-to-face physical meetings and phone calls have been the two most dominant communication methods. Since the rise of the internet, various ways have evolved, from email to social media, installing the mobile application to fill out a form on a website, and eventually waiting for a follow-up. Recently, the increasing use of real-time messaging such as Twitter, Facebook Messenger, WhatsApp, Telegram, Slack, etc., has led to a fundamental transition in how people would prefer to connect with businesses. While most of these communication channels have common characteristics, including online chat, which initially relies only on humans to conduct mutual communication, the baton now is passed to virtual agents or assistants called chatbots. Chatbots, the trendy platform led by virtual assistants, function as customer service representatives who negotiate conversations with clients to improve the user experience and services.

Chatbots are the subsequent major advancement in conversational services, which allow some business companies to communicate through messaging systems, like Twitter and Facebook Messenger, based on artificial intelligence and machine learning. Chatbots can be defined as computer programs living in messenger applications and providing specific services via emulating an interaction with a human through text messaging or a virtual voice [1] [2]. Owing to the overwhelming prevalence of chatbots as messaging is the most commonly used customer assistance medium; therefore, there is a need for the company to invest in a chatbot to support serving their customers' needs as applying in the context of service chatbots. Consequently, companies can strengthen employees' productivity to serve more customers with other services.

Chatbots' primary purpose is to facilitate the conversation between machines and humans in natural language conversation; as in the human viewpoint, these interactions should resemble humans as closely as feasible. Consequently, achieving this has become a fundamental task, with numerous researchers seeking the optimal way for having a chatbot to behave like a human. An effective chatbot should be able to comprehend the user's message, retrieve appropriate information according to the given statement and respond accordingly so that the user perceives the conversation as human-like.

The existing chatbots work just on pattern matching inputs and then finding a scripted answer corresponding to the information presented. The downside to this technique is that it cannot lead to a completely satisfying conversation due to the limitation of discourse within a specific domain with a clear goal. To handle the user's input utterances, Eliza, PARRY, and ALICE, to name a few were among the first chatbots to employ rudimentary parsing, pattern matching, or keyword retrieval approaches. These techniques require handwritten rules to generate responses. Due to the domain-specific

nature of these practices, they were effective at preserving context. However, as the knowledge space expands and users' expectations upsurge, for instance, when engaging in chitchat as in [3], it then becomes difficult to predict users' intentions and considered as not cost-effective, as all the possible patterns must be built manually with a great deal of effort to have a large number of patterns for generating responses.

As artificial intelligence (AI), machine learning, and natural language processing (NLP) techniques advance, researchers and practitioners seek to use data-driven methods incorporating capabilities of deep learning techniques such as RNN, LSTM, and Sequence-to-Sequence (Seq2Seq) model in constructing chatbots automatically and minimizing hand-written rules chatbots techniques. Minimizing the hand-written rules requires the chatbots to be designed based on modular form. This modular form consists of several components: a natural language understanding (NLU) component that turns user text or speech input into semantic representation, an internal state update component that updates the conversation memory (dialogue state tracker), and dialogue policy are used to decide what the following system action will be (dialogue policy), and a natural language generation (NLG) component for producing a response to the user. Training each modular system component typically necessitates a considerable quantity of tagged dialogue data. In contrast to its end-to-end counterparts, the system is more interpretable and stable due to its modular design.

On the other hand, researchers and practitioners recently tried to implement the end-to-end approach utilizing the seq2seq learning task model based on the neural machine translation problem. This attempt is due to the fact that the end-to-end process needs less annotation, giving it an additional viable choice for commercial use cases [4]. In addition, the performance of each component in a modular system is not representative of the entire system because each element is optimized separately [5]. Nevertheless, its end-to-end design makes it uncontrollable [5].

Furthermore, the evolution of machine learning, AI and NLP techniques has encouraged the academics and developers to create chatbots that employ various design strategies. However, despite these advancements in design, chatbots still face several hurdles in comprehending incoming requests, interpreting them, providing acceptable replies, and sustaining a user dialogue. Therefore, academics and developers continue to improve chatbot development techniques to meet the demands of both consumers and service providers. Consequently, it is essential to find a new method to enhance the accuracy of the user utterance's understanding and chatbot's response in service chatbot application. Thus, the fundamental objective and contribution of the current paper aim to present and implement a seq2seq-based learning task model based on encoder-decoder architectural solutions by training generative chatbots on customer support Twitter datasets. These generative chatbots are important to predict automatically an appropriate and automatic response to customers' queries and extensively evaluate their effectiveness in a variety of circumstances under various baseline models, training hyperparameters, and architectures.

The remaining sections of the paper are organized as follows. In Sections II and III, reviews of related works and descriptions of the models are provided, respectively. The methodological approach is presented in Section IV. Section V contains the experimental study of the research, while Section VI includes the conclusion and recommendation for future work.

II. Related Work

Seq2seq learning task models have been implemented in numerous natural language processing tasks, such as chatbot, machine translation, question answering, text summarization, image captioning, sentiment analysis, etc. Initially, seq2seq learning comprising encoder and decoder(E2D) structure was introduced in [6] for Neural Machine Translation (NMT). With the support of gate mechanisms such as LSTM [7] and GRU [8], the problem of vanishing or explosion can be controlled, enabling the model to obtain far longer sentences.

To better capture the dependencies in utterance, bidirectional and reverse order practices are commonly used to design the seq2seq models [9] [10]. Yet, at the same time, this approach also has a fixed-length vector (context vector) issue. This issue arises in the decoding process because the source sentences will compress the input regardless of the length vector this neural network needs as a context vector, especially when the source sentence is long [8]. Indirectly, this process leads to incorrect responses, as each word in the answer may have a close relationship with various sections of the words in the request.

A study was done in [11] and [12] combated the problem by adding an attention mechanism layer and integrating it into the decoder by repeatedly reading the representation of a source sentence, which remains fixed after being generated by the encoder. Hence, the model is able to search for relevant parts to predict a targeted word and attain cutting-edge machine translation performance. Inspired by the great commission in machine translation, the researchers and developers attempt to apply this technique to other tasks, including chatbots. For instance, in [13], the author also used seq2seq learning by comparing the performance of chatbot-generated responses between LSTM, GRU, and Convolution Neural Network (CNN). Unlike the above approaches, our work integrates an additive attention model to align the relevancy of input and output for improving question-answer. In addition, a study in [14] also employed attention mechanisms to the encoder-decoder architecture in enhancing question-answer relevance.

In [15], the authors implemented an attention-based with encoder-decoder neural architecture with the knowledge graph, and the corpus joins embedding as input in a task-oriented based chatbot. In contrast to [16], the authors added information regarding the conversation history and external knowledge collected from the search engine to enhance the seq2seq chatbot. In [17], the authors allocated labeled data to hierarchical categories using the attention-based Seq2Seq model. In the research, when answer predictions were inconsistent, a slot-filling method was used to determine which questions needed to be asked in order to make correct predictions.

All the above-related works focus less on examining the effects of complementary mechanisms in deep learning, such as batch size, lstm size or types of embedding in different seq2seq chatbot architecture. Therefore, this work attempts to

focus more on investigating such effects through experimental study as presented in Section V.

III. Model Description

This section describes and explores our model architecture based on the word representation model, seq2seq RNNs (LSTM, GRU, biLSTM, biGRU), and an attention mechanism.

A. Word Representation Model

For a computer to comprehend the meaning of the words and sentences, the text data must be converted to a numerical format. Embedding (encoding or vectorizing) is the term used for this concept. Character embedding, word embedding, and phrase embedding are only a few examples of many other types of embedding. Among various types of embedding, the word embedding is most commonly employed [18]. Word embedding is a way to model language that maps words to vectors of real numbers. It encodes words or sentences with several dimensions in vector space. The embedding layer can be initialized using pre-trained word vectors such as Word2vec, Glove, or FastText as implemented in this research word. A detailed description of the word embedding models is presented in the following subsection:

1) Word2Vec: Word2vec is a predictive embedding technique that uses the low dimensionality of word vectors to learn fine word vectors from massive data sets containing billions of words. There are two main architectures of Word2Vec for producing a distributed word representation, namely:

- Continuous bag-of-words (CBOW) model
  - This architecture is based on the language model of a feed-forward neural network [19]. It seeks to anticipate the current word based on the surrounding context by minimizing the loss function
- Skip-Gram model
  - Unlike the CBOW model, this model is aimed to predict surrounding words given the current word.

2) Glove: The Global Vectors for Word Representation (GloVe) enhances the Word2vec approach proposed in [20] at Stanford in 2014 for effectively learning word vectors. Conceptually, the Word2vec approach only considered local contexts but did not utilize a global context. Previously, the conventional vector space model representation of words was built using matrix factorization techniques such as Latent Semantic Analysis (LSA), which gave a better result than global text statistics. This model is also called a count-based model. Count-based models learn their vectors by reducing the co-occurrence counts matrix's dimensionality. However, this technique does not give a promising result as a learned method or predictive model such as Word2Vec captures the meaning and performs an arithmetic operation that can pose semantic or syntactic relationship of words, for example, $king - man + woman \rightarrow queen$. Thus, by merging the global matrix factorization and local context window approaches with the help of a bilinear regression model, GloVe indirectly benefits from both techniques.

3) FastText: FastText, made available by Facebook, Inc, is one of the contributions to prediction-based word embedding models. The usually cited work for this model is from [21] and [22]. The motivation behind the FastText model is because of the shortcoming of word embedding models that disregard the word's morphology and learn a distinct vector for each word.

The improvement is made based on the Skip-Gram model introduced in [23], wherein each word is represented as a bag of character n-grams. Each word is mapped to a set of n-grams, and the skip-gram model is modified to regard each word vector as the sum of its n-grams, which is based on the assumption that similar groups of letters express identical meanings.

Since word vectors are composed of known n-grams, hence can also be computed on unknown words. Consequently, even a word not in the vocabulary is assigned a vector based on its subword units. This unknown term is even more essential for inflected languages, as some inflected forms of words are uncommon and may not even be present in the training set. Training the FastText embedding is faster than the majority of other options due to their simplicity and efficient implementation.

B. Sequence-to-Sequence (seq2seq) Learning Task Model

Seq2seq learning task was initially proposed in machine translation for training models to map the sequence of input between one domain (e.g., German sentences) to output sequences in a different domain (e.g., the same sentence translated to English). Due to its promising result, many researchers investigated and worked by adopting this technique in various tasks such as image captioning, text summarization, and chatbot (question and answering task). The bot generates a natural language response as an output sequence given a natural language question as an input sequence.

The most typical architecture for constructing a seq2seq learning problem is using the encoder-decoder architecture. This architecture in the seq2seq learning task manifested in three parts: encoder, context vector (final hidden/internal state vector), and decoder, as the name implies. The encoder attempts to convey the meaning of the input sentence by encoding it into a fixed-size hidden representation. This hidden representation is converted to output by a decoder. The fundamental structure of this model is based on two RNNs (or can be used as another type of RNNs such as LSTM/GRU for better performance) [6]. Encoding the input into a vector representation employs one RNN as the encoder that captures the context and essential information of the input sequence. On the other hand, the other RNN (as a decoder) will then take this vector as input and use it to generate the output sequence. The basic architecture of the seq2seq model in training mode is illustrated in Fig. 1.

Based on this figure, let $x = \{x_1, x_2, x_3, ..., .x_n\}$ represent the words contained in each input statement or utterance (where n being the statement's length) is mapped in the form of embedded representation (φ^{x^n}) and passed to a variant type of RNN models such as LSTM or GRU. The embedded representation can be either pre-trained embedding such as Word2Vec, Glove, and FastText or jointly trained during model training to convert words into dense vectors, as mentioned

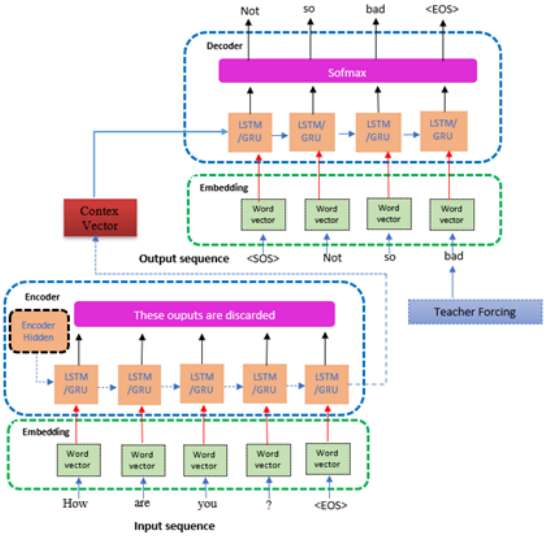

Fig. 1. Basic Encoder-Decoder Architecture While Training.

in the previous section. RNNs also take the first encoder hidden state as the input. RNNs take all these embeddings and sequentially give hidden representation and output vectors at each time step. Hence, it takes a word and a hidden state of the previous state as an input and provides one output and updated hidden state until it reaches the end of the input mark with a unique token known as <EOS>, the outputs at each time step of the encoder part are all discarded since outputs will be summarized by the context vector (C). This context vector contains information about all of the input items that enable the decoder to predict accurately. Equations (1) and (2) illustrate the computation of hidden states and context vectors, respectively;

$$h_m = f_1(\varphi^{x^n}, h_{m-1}) \tag{1}$$

$$c = f_2(\{h_0, h_1, ...h_M\}) \tag{2}$$

where h denotes as the hidden state, c denotes the context vector constructed from the encoder hidden states and f_1, f_2 are nonlinear functions such as LSTM / GRU in this case.

The context vector also acts as the decoder's initial hidden state to pass information from the encoder to the decoder. Before passing to the decoder, this work considers uni-directional or bi-directional encoders, wherein bidirectional encoder; there will be one forward RNN and one reverse RNN. The processing of an input sequence occurs in both directions (forward and backward). The forward and backward hidden states are then combined before being transmitted to the decoder.

In the decoding phase, at the first timestep, the <SOS> token is given as input to RNNs along with the context vector. <SOS> marks the beginning of decoding, and it generates the first word of the chatbot response by looking at the context vector. This decoding first generated output of RNNs probably *"Not"*. For the next timestep, the *"Not"* will be given as the input along with the previous timestep hidden state. This step will provide an output as *"so"*. This output generation will continue until it reaches a unique token known as <EOS> is encountered. Considering the context vector as c, and all previously predicted output as $\{y_1, y_2, y_3, ...y_{t-1}\}$, the decoder has been trained to anticipate the following token y_t. This prediction is the maximum likelihood estimation of y_t. The prediction is given y, the output vector, and c, the context vector. Thus, the $p(y)$ is computed as in Equation (3):

$$p(y) = \prod_{t=1}^{T} p(y_t|y_1, y_2, y_3, ...y_{t-1}, x_t) \tag{3}$$

and produces a token with a conditional probability for each timestep t through the following Equation (4):

$$p(y_t|y_1, y_2, y_3, ...y_{t-1}, x_t) = g(y_{t-1}, s_t, c_t) \tag{4}$$

where $g(.)$ is a softmax function and s_t is the decoder's hidden state at the timestep t which can be computed as in the Equation (5) as follows:

$$s_t = f(s_{t-1}, y_{t-1}, c_t) \tag{5}$$

1) Long Short Term Memory (LSTM): LSTM was developed as a short-term memory solution and initially proposed in [7]. LSTMs are a type of RNN that uses specific hidden states to manage long-term dependencies better while memorizing inputs over time [7]. The difference from standard RNN is how the hidden state is calculated within LSTM cells. LSTMs architecture has an internal cell state that acts as a transport highway that can carry and filter the information in a sequence by adding or removing it. The adding and removing information state is controlled by a structure known as gates.

Gates have sigmoid activations identical to the tanh activation, whereas rather than squishing values from -1 to 1, it squishes values from 0 to 1. This value is significant when updating or removing the data as every value multiplied by 0 is 0, allowing the values to be vanished or be 'forgotten'. On the other hand, every value multiplied by one will result in the same value, remaining unchanged or 'preserved'. Subsequently, the network learns which information is irrelevant and may be removed and which information should be retained with this activation function. Three more gates are generally added in contrast to standard RNNs cells, namely, forget, input, and output gates, as shown in Fig. 2.

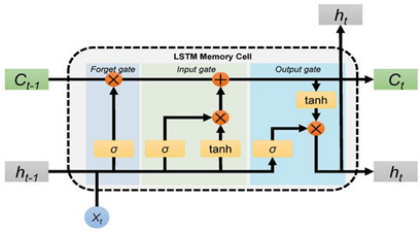

Fig. 2. The Long Short-Term Memory Introduced in [7].

2) Gated Recurrent Units (GRU): GRUs is a more recent generation of RNN cells compared to an LSTM introduced in [8]. The author has introduced two gates: the update gate and the reset gate, as depicted in Fig. 3. The update gate combines the forget and input gates in GRUs and operates similarly to LSTMs, controlling what information to discard and add. On the other hand, the reset gate is a different gate used to

specify the amount of prior information that can be discarded. In GRUs, it also merges the hidden state and cell state, and thus, the output gate is no longer needed. Therefore, this model is simpler while gaining more popularity than regular LSTM, with fewer parameters and faster training than LSTM [8].

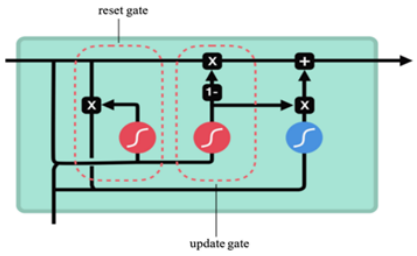

Fig. 3. Types of Gates in GRU Cell Introduced in [8].

3) Bi-Directional RNNs Cell: The previous section discusses the implementation of uni-directional RNNs, which means it runs in a single direction. This research attempts to employ bidirectional RNNs (both for LSTM and GRU) to improve model performance by incorporating past and future context information. This bidirectional means each layer has two RNNs: one running in the forward direction of the sequence (from left to right) and another running in the backward direction (from right to left) to capture dependencies in two contexts [24]. The resulting forward and backward outputs are concatenated before being passed on to the next layer, as shown in Fig. 4. The encoded representation of each word now has the information of the reverse and the future words of the particular word to predict output better.

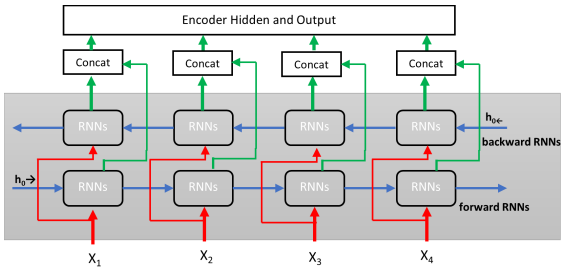

Fig. 4. The Bi-Long Short-Term Memory (biLSTM) Process Captures Sequential Features. The Last Hidden Layer of the LSTM is Extracted as Features Representing the Text.

4) Neural Attention Mechanism: The previous section describes the seq2seq model based on RNNs. These RNN structures utilize the temporal dynamics of the input data to generate sequential output data. However, the output created at a particular timestep and the input sequence used to obtain that result may or may not be relevant remains uncertain. Moreover, in the seq2seq model using RNNs, all the intermediate states of the encoder output will be discarded, and only its final states (context vector) will be used to initialize the decoder. This technique incorporates well with short or medium sequences; however, as the lengths of the sequence grow, a single vector becomes congested and more challenging to analyze long sequences into a single vector.

Meanwhile, RNNs sequentially process tokens while preserving a state vector representing the data observed after each token. The information from the inputs can be arbitrarily propagated in the sequence through the continuous encoding of the data. Due to the vanishing gradient problem, the model's state towards the end of a long sentence typically does not have information about earlier tokens. As a result, the process does not perform as expected. Long sequences benefit from LSTM, reducing disappearing and exploding gradient effects, albeit not entirely eliminated. Furthermore, RNN architectures may be unable to handle increasingly complex feature representations in order to produce reliable outputs.

The aforementioned issue was resolved with the introduction of attention mechanisms introduced in [11] and [12]. Attention processes enable a model to directly examine the condition of an earlier point in the sentence and derive conclusions from it. The attention layer has access to all previous states. It can weigh them according to some learned measure of relevance to the present token, allowing it to provide more precise information on distant relevant tokens, as illustrated in Fig. 5. It decides which source elements are the most important at each decoder step. The encoder does not need to condense the whole source into a single vector in this case; instead, it provides token representations for all of the source data (for example, all RNN states instead of the last one). In addition, the key concept behind attention is not to throw away these intermediate encoder states but to make use of all the states to create the context vectors that the decoder uses to produce the output sequence through attention weight. The attention weight is computed to decide which part of the input was relevant and subsequently determine the output.

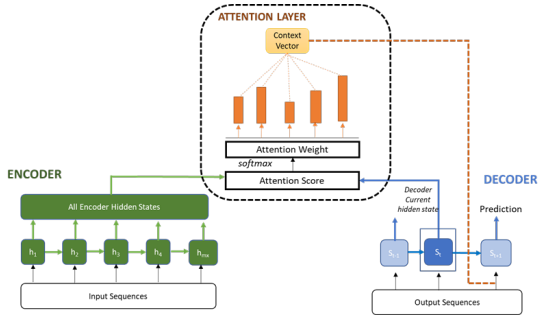

Fig. 5. Attention-Based E2D.

In the attention process, the relevance of each word in the input sequence will be determined for each output cell. For each y_t in the output y, it is influenced by the context vector c_t (source context for decoder step t) are used in an information filter for all hidden states $h = \{h_1, h_2, h_3, ..., h_{m_x}\}$ of the encoder, which can be computed as in the following Equations (6), (7) and (8):

$$c_t = \sum_{i=1}^{m_x} \alpha_{t_i} h_i \tag{6}$$

Where α_{t_i} is calculated by

$$\alpha_{t_i} = \frac{exp(e_{ti})}{\sum_{j=i}^{m_x} exp(e_{tj})} \tag{7}$$

Where $e_{ti} = align(s_{t-1}, h_i)$ refers to the additive score function that considers

$$e_{ti} = V_a^T tanh(W_a s_{t-1} + U_a h_i) \quad (8)$$

Where α_{t_i} indicates the attention weights that the model has learned, W_a ,U_a and V_a, implies another weight parameter for the model to learn. The align is an alignment model for evaluating the relationship between the input of position i and the output of the position t.

IV. Methodological Approach

This section gives an overview of the current research methodological approach to previously discussed implemented models. Fig. 6 depicts the methodology steps involved in this work as follows:

- *Dataset Preparation* – The dataset is collected using publicly available datasets from free online websites which we examine the data and explore the dataset using some fundamental Exploratory Data Analysis (EDA).
- *Data Prepossessing* – Load the text, perform pre-processing or data cleaning, and do a train-test split. In this phase, we build questions-answer pair. Append <START> and <END> to all the answers. Create a Tokenizer and load the whole vocabulary into it with the help of embedding techniques for feature extraction.
- *Modeling* – Define the model. We implement an encoder-decoder architecture-based seq2seq learning task model with and without attention to a different variant of RNN cells
- *Training and tuning* – The model will be trained and tuned with the help of various hyperparameter optimization and regularization techniques to overcome overfitting during training. We aim to minimize the objective function during training by reducing the loss.
- *Results* – The trained model will be evaluated using a valid/test set through BLUE score and validation loss during training based on predicting answers.

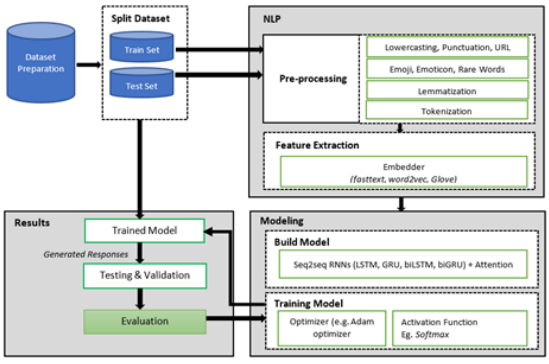

Fig. 6. Illustrates the Methodology Steps.

V. Experimental Study

This section presents different experiments conducted to study the functionality of various models employed (as mentioned in the previous section) and thoroughly examine their performance. The experimental results are compared based on the effect of several variables such as the encoder types, adding an attention layer, variant of embedding, the number of hidden sizes, and batch sizes. The dataset used in the experiment, experiment settings, evaluation methods, and qualitative analysis of experimental results are described in the following subsections. before

A. Datasets

The experiment trains and evaluates the models using the *"Customer Support on a Twitter (CST)"* dataset from Kaggle[1]. The CST dataset was collected in 2017 with a huge, innovative corpus of tweets and replies for the advancement of NLU and conversational models, along with research into modern customer service techniques and impact. It consists of 2,811,774 tweets and replies, with 1,537,843 (54.69%) tweets generated from consumers and 1,273,931 (45.31%) generated from customer supports agent. Among these 1.5 million customer tweets, about 1.27 million received replies from customer support agents, and 0.23 million otherwise. One of the main reasons to use this dataset is that it contains real-life conversations between customers and customer support agents with natural responses from support agents for accurately explaining problems and solutions. Moreover, it is practical since it permits a relatively small message size restriction for recurrent networks.

While conducting an exploratory analysis of the dataset, it can be observed that there are 108 customer support brands represented in the dataset, and 597075 consumers' requests are answered. The top 20 customer support replies related to the company brand are depicted in Fig. 7.

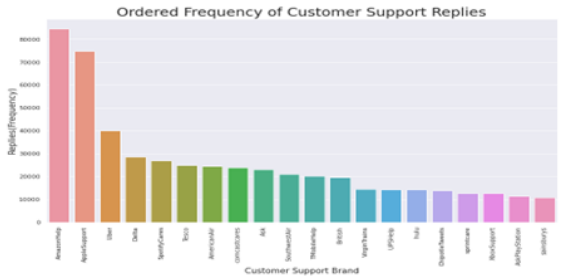

Fig. 7. The Top 20 Customer Service Replies by Brand.

In addition to this observation, it was identified that Amazon's customer service responded to a large number of inquiries, followed by Apple and Uber. There are a lot of companies in the dataset that have minimal responses or had no responses at all.

As shown in Table I, the information in the dataset must be restructured to create a conversational dataset between consumer and customer support agents suited for the current

[1]https://www.kaggle.com/datasets/thoughtvector/customer-support-on-twitter

TABLE I. DATASET FEATURES DESCRIPTION

Features	Description	Datatypes
tweet_id	The unique ID for this tweet	Int64
author_id	The unique ID for this tweet author (anonymized for non-company users).	Object
Inbound	Whether or not the tweet was sent (inbound) to a company on Twitter. This feature is useful when re-organizing data for training conversational models.	Bool
created_at	Date and time when the tweet was sent.	Object
Text	The text content of the tweet.	Object
response_tweet_id	IDs of tweets that are responses to this tweet.	Object
in_response_to_tweet_id	IDs of the tweet is in response to, if any.	Float64

study. In the restructuring process, the dataset features must be filtered by selecting only inbound tweets that are not retweeted. Then, apply the *"in_response_to_tweet_id"* and *"tweet_id"* features to associate each tweet with the relevant reply based on the inbound feature, excluding instances where response tweets are not from a company. As the data is still in unstructured text data, additional preprocessing is required to eliminate unnecessary features such as emoji and emoticons, lower casting, non-English tweets, etc. The new dataset has 794,299 rows and six columns consisting of *'author_id_x'*, *'created_at_x'*, *'text_x*, *author_id_y'*, *'created_at_y'*, and *'text_y'*, where *x* and *y* are represented as a question from consumers and answered by customer service agent respectively.

Training and validation are conducted independently on 75% and 35% of the entire dataset. The model is termed accurate if the predicted response matches the ground-truth answer. This current research incorporates the Bilingual Evaluation Understudy (BLEU) score function to evaluate the performance models.

B. Experimental Settings

The study compares the attention-based approach to related baseline models that do not employ attention mechanisms to evaluate how well the models work. These models are implemented in a python-dependent package on a deep neural network framework called TensorFlow [25] and Keras[2]. We trained models on a GPU with 3082 CUDA cores and a VRAM of 12GB. The model is trained for 500 epochs (a high value is set since the study employs the early stopping technique) and tested on a batch size of 64, 128, 256, and 304 (the number can be divisible by 8). While the hidden size of LSTM and GRU is tested on 100, 200, and 300 units. For the optimization, the study uses the Adam optimizer with a learning rate of 0.003 [26]. A gradient clipping of 50.0 is implemented to combat the 'exploding gradient' problem, preventing the gradients from expanding exponentially and causing the cost function to either overflow (with undefined values) or overshoot cliffs. All the weights and biases are initialized using Xavier and glorot uniform distribution [27].

This study uses 300-dimensional pre-trained word embeddings for Fasttext[3], GloVe[4], and Word2Vec[5]. An early stopping technique with patience four is adopted to combat overfitting. Table II shows the hyperparameters and their corresponding ranges for training the models. It also presented the best-performing hyperparameter for each of the models.

TABLE II. HYPERPARAMETER SETTING

Parameter	Range	Final Setting
Max input Length	39	39
Word embedding	FastText/Glove/Word2vec	FastText/Glove
Embedding size	300	300
Encoder types	Unidirectional/Bidirectional	Bidirectional
Learning Rate	0.003	0.003

C. Performance Evaluation Metrics

Following [13] and [15], this current study adopts BLEU as suggested in [28] best to evaluate the performance of our model. According to the definition provided in [28], BLEU evaluates the co-occurrences of n-grams in the reference human translation and recommended answers. It computes the n-gram precision for the entire dataset, compounded by a brevity penalty to penalize brief translations. The more closely a machine translation resembles a professional human translation, the better.

The BLEU score compares the chatbot-generated output text (hypothesis) to a human-generated response text (reference). It specifies how many n-grams from the output text are included in the reference. The BLUE score can take on any value within the interval [0, 1] and is technically defined as Equation (9).

$$BLEU = BP \times \left[\prod_{n=1}^{N} precision_n\right]^{1/N} \tag{9}$$

Where N is the maximum n-gram number $n = [1, N]$ (N=4 in our evaluations). BP (brevity penalty) and $precision_n$ are defined with Equation (10) and (11), respectively.

$$BP = min(1, exp(1 - \frac{ref_{lenght}}{out_{lenght}})) \tag{10}$$

Where ref_{lenght} is reference length, and out_{lenght} is the chatbot output length

$$precision_n = \frac{\sum_n min(m^n_{out}, m^n_{ref})}{\sum_{n'} m^{n'}_{out}} \tag{11}$$

Where m^n_{out} is the number of n-grams matching the reference in the chatbot output, m^n_{ref} denote as the number of n-grams in the reference, and $\sum_{n'} m^{n'}_{out}$ implies the total number of n-grams in the output of the chatbot.

The BLEU scores were computed using the blue score module from the translated package on the nltk[5] platform, which was built in Python.

[2]www.keras.io.

[3]http:fasttext

[4]http://nlp.stanford.edu/projects/glove/

[5]https://code.google.com/archive/p/word2vec/

[5]Natural Language Toolkit: Available online: https://www.nltk.org/

D. Results and Comparison

This section elaborates on the experimental results from our model on the mentioned dataset. The experiment assessed the performance of the different models based on encoder types and various types of embedding with varying parameters by comparing models to baseline RNNs such as LSTM and GRU. After multiple experiments, the study concluded the result of the promising hyperparameter and settings as presented in Table II. One of the models uses a pre-trained 300-dimensional word embedding as a hidden size tested on varying numbers of 100, 200 300, where the hidden size of 300 gives a promising result of a pre-trained 300-dimensional word embedding.

The experimental results on all the models indicated that the bidirectional encoder type attention-based models achieved promising performance and outperformed the neural networks that do not use an attention mechanism as a baseline. The different hyperparameters are used to test each model, and the promising performing hyperparameters for each model are shown in Table II. The impact of some of these hyperparameters will be highlighted, and an example of generated response based on several models is presented in the following subsections.

1) Effect of Encoder Type in E2D Architecture: This section evaluates the performance of the two distinct encoder types used in these models. In comparison to a uni-directional encoder, the model performance with bi-directional encoder types yields promising results, as shown in Fig. 8. This situation might happen because the bi-directional encoder works by preserving the information from both sentence directions, allowing the network to predict the next word better as it can understand the sentence context more. Thus, the bi-directional encoder is promising to be used for further analysis in this study.

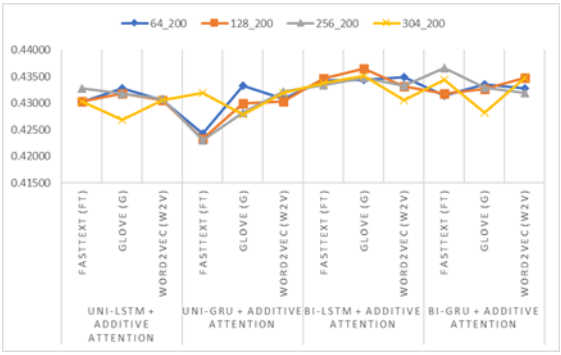

Fig. 8. E2D Network of RNNs with Different Encoder Types and Embeddings.

2) Effect of Attention Layer in E2D Architectures: This section uses hyperparameters from Table II to evaluate the RNNs E2D model with and without attention. As depicted in Fig. 9, the model's performance with attention vastly outperforms the RNNs E2D model without attention on this dataset. Therefore, the attention-based models improve the predictive performance of the Seq2Seq chatbot models. Furthermore, in comparing the performance of the RNNs types, the LSTM gives promising results in almost all models, as shown in Fig. 8. However, the GRU model can be an option if computational time is considered since GRU trains faster than LSTM, and the result for this dataset is not much different than LSTM.

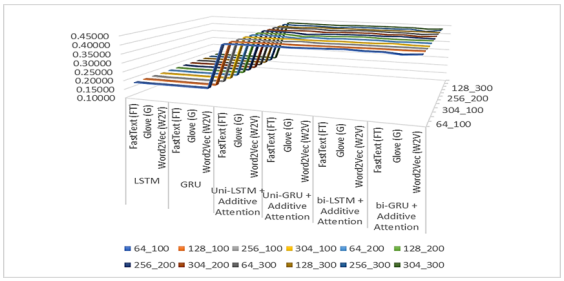

Fig. 9. Encoder-Decoder Network Model with and without Attention.

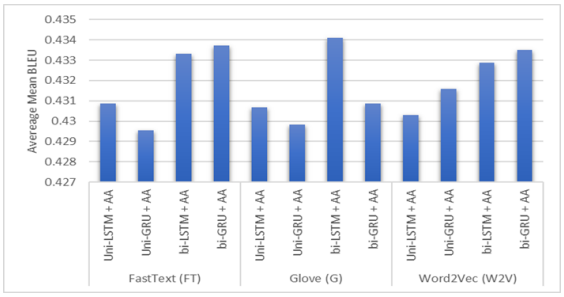

Fig. 10. Different Embedding Types in Encoder-Decoder Architecture.

3) Effect of Different Embedding Types in Encoder-Decoder Architectures: The performance of the RNNs models with varying types of embedding (FastText, Glove, and Word2Vec) is analyzed by finding the BLEU score among different variants of the models. As shown in Fig. 10, FastText and Glove embedding types give better performance for this type of chatbot dataset. Moreover, these embedding types perform better when integrating with a bi-directional encoder with LSTM (for Glove embedding) or GRU (for FastText).

4) Effect of Hidden Sizes in E2D Architectures: The performance of RNNs encoder-decoder models with different hidden sizes/units is evaluated in this section. The hidden size parameter is tested on 100, 200, and 300 units with a fixed embedding size of 300, as in Table II. Most models produce better results for 200 or 300 units, where 42% of each out of total models occur, as presented in Table III. Since a fixed embedding size of 300 is used, the good hidden size is preferable to be an equal size with the embedding size as indicated in this experiment result for this dataset.

5) Effect of Batch Sizes in E2D Architectures: The different batch sizes evidenced in this experiment were $B = [64, 128, 256, 304]$ with hyperparameter setting as in Table II. As illustrated in Fig. 10, the smaller batch size of 64 produces better results amongst models. Furthermore, the batch size of 64 with 300 hidden sizes produced more promising results in this experiment dataset, indicating best used for further analysis in this research. The smaller batch sizes correlate to stability during the training and are better for accuracy. On the other hand, the larger batch size is better for computational speed. Nevertheless, batch size should be adequate so that the data would fit into memory. Due to the limitation of memory

space, the experiment can be done with a maximum batch size of 304.

VI. Qualitative Analysis

To qualitatively comprehend the model's performance gained with a different experiment configuration, as shown in Table II, the study prepares the outputs responses for a specific customer inquiry. The customer's query may be based on an emotive or informative question. For the informative type customer query instances, *"When is delta is last scheduled flight?"*, predicted responses are given in Table III. On the other hand, Table IV shows an example of an emotional types question.

As observed, most RNNs models' response prediction can provide general and reasonable answers for customer queries. However, both types of questions can predict better responses by implementing bi-directional RNNs based on the actual or target response. Based on the emotional type query, the bidirectional with attention models can predict the specific word, the *"account"*, rather than *"details"* (a general response but acceptable) as referred to in the actual reaction. Moreover, the models are more emotional in responding to customer queries than humans for both types of questions. Conversely, the informative queries and asking for particular information requests are difficult to formulate, and the resulting responses are less pertinent to the question. However, by looking at the answers of various models, it was discovered that bidirectional LSTM with attention generates better logical statements, and the response appears to be indistinguishable from the actual one.

TABLE III. Example of Generated Response for Informative Type Question (64_300 FastText)

Customer Questions	When is delta is last scheduled flight?
Seq2seq E2D LSTM	hi there i am sorry to hear that you are having a problem with the app please do your confirmation number i will be happy to help tab
Seq2seq E2D GRU	i am sorry to hear that you are having trouble with your flight please do your confirmation number so i can take a look at your flight tow
Seq2seq E2D LSTM Unidirectional + Attention	hi there we are not showing any announcements on this flight if you have any other questions please let us know by
Seq2seq E2D GRU Unidirectional + Attention	i am sorry for the delay in your flight i am showing that you are not able to get a gate to clear your flight tow
Seq2seq E2D LSTM bidirectional + Attention	i am sorry to hear that you are not able to get through to us please do me your confirmation number and i will gladly look into it for you ¡end¿
Seq2seq E2D GRU bidirectional + Attention	hi there we are not sure what you mean we are here to help please do us your confirmation number so we can follow up ¡end¿
Actual	**certainly as soon as we get a response from the appropriate team that has the info we will be sure to tweet you**

VII. Conclusion and Future Work

As a conclusion, we attempt to construct a generative service chatbot to automatically respond to customers' queries for assisting companies in having a 24-hour support system. To construct this system, seq2seq learning task models based on encoder-decoder architectures (with and without attention as the baseline) are deployed. This study further investigates these models through different RNNs structures (LSTM and GRU), encoder types (uni-directional and bidirectional), and different embedding types (FastText, Glove, and Word2Vec) and tested with varying parameters of training.

TABLE IV. Example of Generated Response for Emotional Type Question (64_300 FastText)

Customer Questions	you have lied gave me the run around have the worst customer service ever so disappointed i cant work without internet
Seq2seq E2D LSTM	i am sorry to hear that you are having trouble with your internet service i would be happy to help please do me the details of your issue
Seq2seq E2D GRU	i am sorry to hear that you are having trouble with your internet service i would be happy to help please do me the details of your issue
Seq2seq E2D LSTM Unidirectional + Attention	i apologize for the issues you are having with the services have your concerns been addressed if not i will be
Seq2seq E2D GRU Unidirectional + Attention	i am sorry for the poor experience can you please do pm the full service address and name on the account as we
Seq2seq E2D LSTM bidirectional + Attention	i am sorry to hear about the poor experience can you please do the full service address and name on the account as we assist
Seq2seq E2D GRU bidirectional + Attention	i am sorry to hear that you are having issues with your internet services please do your account details to help
Actual	**it is unfortunate you are having trouble to better assist please do me the account number thank you ami**

Based on the experimental results, the bi-directional RNNs (LSTM and GRU) attention mechanisms produced promising results for further work in a chatbot. Moreover, there are not many effects typed of embedding as well as batch size and hidden sizes for this dataset since the result is not significantly different. However, the stability of the result is obtained from batch size 64, with 300 hidden sizes (as the same value with embedding size) combining with FastText or Glove can be opted for further work. Additionally, based on the findings for this dataset, it is proven that while biLSTM performs better with Glove, biGRU operates better with FastText. Overall, adding an attention layer improved the BLEU score compared to the baseline models.

The significant of findings from this research work is that the proposed attention mechanism has demonstrated its ability to meet the aforementioned requirements. Therefore, this proposed work provides an extension of the existing technique in developing chatbots. The attempt is made through end-to-end approaches by using seq2seq learning task model adoption from neural machine translation. A potential direction for future research can be explored and examine the variant of attention mechanism based on different scoring functions in comparing to the current experimental results and findings. In addition, the experiment also can be investigated in other deep learning architectures such as generative adversarial neural network and training parameters, including varying learning rates, dropout rates, optimizer, and activation functions.

Acknowledgments

This research is partly funded by Ministry of Higher Education Malaysia under grant R.J130000.7851.4L942. The authors would also like to thank the Universiti Malaysia Sarawak (UNIMAS) and Universiti Teknologi Malaysia (UTM) for providing the resources used in this research work.

REFERENCES

[1] E. Adamopoulou and L. Moussiades, "An overview of chatbot technology," *Artificial Intelligence Applications and Innovations*, vol. 584, pp. 373 – 383, 2020.

[2] M. F. McTear, "The rise of the conversational interface: A new kid on the block?" in *Future and Emerging Trends in Language Technology. Machine Learning and Big Data - Second International Workshop, FETLT 2016, Seville, Spain, November 30 - December 2, 2016, Revised Selected Papers*, ser. Lecture Notes in Computer Science, J. F. Quesada, F. Martín-Mateos, and T. López-Soto, Eds., vol. 10341. Springer, 2016, pp. 38–49. [Online]. Available: https://doi.org/10.1007/978-3-319-69365-1_3

[3] Z. Yan, N. Duan, J. Bao, P. Chen, M. Zhou, and Z. Li, "Response selection from unstructured documents for human-computer conversation systems," *Know.-Based Syst.*, vol. 142, no. C, p. 149–159, feb 2018. [Online]. Available: https://doi.org/10.1016/j.knosys.2017.11.033

[4] Z. Zhang, R. Takanobu, M. Huang, and X. Zhu, "Recent advances and challenges in task-oriented dialog system," *CoRR*, vol. abs/2003.07490, 2020. [Online]. Available: https://arxiv.org/abs/2003.07490

[5] J. Gao, M. Galley, and L. Li, "Neural approaches to conversational ai," 2018, cite arxiv:1809.08267Comment: Foundations and Trends in Information Retrieval (95 pages). [Online]. Available: http://arxiv.org/abs/1809.08267

[6] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, 2014, pp. 3104–3112. [Online]. Available: https://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] K. Cho, B. van Merrienboer, Ç. Gülçehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, A. Moschitti, B. Pang, and W. Daelemans, Eds. ACL, 2014, pp. 1724–1734. [Online]. Available: https://doi.org/10.3115/v1/d14-1179

[9] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," in *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, ser. AAAI'16. AAAI Press, 2016, p. 3776–3783.

[10] A. Xu, Z. Liu, Y. Guo, V. Sinha, and R. Akkiraju, "A new chatbot for customer service on social media," in *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems*, ser. CHI '17. New York, NY, USA: Association for Computing Machinery, 2017, p. 3506–3510. [Online]. Available: https://doi.org/10.1145/3025453.3025496

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

[12] T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, L. Màrquez, C. Callison-Burch, J. Su, D. Pighin, and Y. Marton, Eds. The Association for Computational Linguistics, 2015, pp. 1412–1421. [Online]. Available: https://doi.org/10.18653/v1/d15-1166

[13] M. Aleedy, H. Shaiba, and M. Bezbradica, "Generating and analyzing chatbot responses using natural language processing," *International Journal of Advanced Computer Science and Applications*, 2019.

[14] C. Xing, W. Wu, Y. Wu, J. Liu, Y. Huang, M. Zhou, and W.-Y. Ma, "Topic aware neural response generation." in *AAAI*, S. P. Singh and S. Markovitch, Eds. AAAI Press, 2017, pp. 3351–3357. [Online]. Available: http://dblp.uni-trier.de/db/conf/aaai/aaai2017.html{#}XingWWLHZM17

[15] F. Kassawat, D. Chaudhuri, and J. Lehmann, "Incorporating joint embeddings into goal-oriented dialogues with multi-task learning," in *European Semantic Web Conference*. Springer, 2019, pp. 225–239. [Online]. Available: https://jens-lehmann.org/files/2019/eswc_jointembedding_dialoguesystems.pdf

[16] Z. Wang, Z. Wang, Y. Long, J. Wang, Z. Xu, and B. Wang, "Enhancing generative conversational service agents with dialog history and external knowledge," *Comput. Speech Lang.*, vol. 54, pp. 71–85, 2019. [Online]. Available: https://doi.org/10.1016/j.csl.2018.09.003

[17] M. Patidar, P. Agarwal, L. Vig, and G. Shroff, "Automatic conversational helpdesk solution using seq2seq and slot-filling models," in *Proceedings of the 27th ACM International Conference on Information and Knowledge Management, CIKM 2018, Torino, Italy, October 22-26, 2018*, A. Cuzzocrea, J. Allan, N. W. Paton, D. Srivastava, R. Agrawal, A. Z. Broder, M. J. Zaki, K. S. Candan, A. Labrinidis, A. Schuster, and H. Wang, Eds. ACM, 2018, pp. 1967–1975. [Online]. Available: https://doi.org/10.1145/3269206.3272029

[18] S. Mohamad Suhaili, N. Salim, and M. N. Jambli, "Service chatbots: A systematic review," *Expert Syst. Appl.*, vol. 184, no. C, dec 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.115461

[19] Y. Bengio, R. Ducharme, P. Vincent, and C. Janvin, "A neural probabilistic language model," *J. Mach. Learn. Res.*, vol. 3, pp. 1137–1155, Mar. 2003. [Online]. Available: http://dl.acm.org/citation.cfm?id=944919.944966

[20] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation." in *EMNLP*, vol. 14, 2014, pp. 1532–1543.

[21] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," 2016, cite arxiv:1607.04606Comment: Accepted to TACL. The two first authors contributed equally. [Online]. Available: http://arxiv.org/abs/1607.04606

[22] A. Joulin, E. Grave, P. Bojanowski, M. Douze, H. Jégou, and T. Mikolov, "Fasttext.zip: Compressing text classification models." *CoRR*, vol. abs/1612.03651, 2016. [Online]. Available: http://dblp.uni-trier.de/db/journals/corr/corr1612.html{#}JoulinGBDJM16

[23] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[24] P. Zhou, Z. Qi, S. Zheng, J. Xu, H. Bao, and B. Xu, "Text classification improved by integrating bidirectional lstm with two-dimensional max pooling." in *COLING*, N. Calzolari, Y. Matsumoto, and R. Prasad, Eds. ACL, 2016, pp. 3485–3495. [Online]. Available: http://dblp.uni-trier.de/db/conf/coling/coling2016.html{#}ZhouQZXBX16

[25] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard *et al.*, "Tensorflow: A system for large-scale machine learning." in *OSDI*, vol. 16, 2016, pp. 265–283.

[26] L. Mou and Z. Jin, *Tree-Based Convolutional Neural Networks: Principles and Applications*, 1st ed. Springer Publishing Company, Incorporated, 2018.

[27] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks." in *AISTATS*, ser. JMLR Proceedings, Y. W. Teh and D. M. Titterington, Eds., vol. 9. JMLR.org, 2010, pp. 249–256. [Online]. Available: http://dblp.uni-trier.de/db/journals/jmlr/jmlrp9.html{#}GlorotB10

[28] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: a method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.