Deep RL Conversational Chatbot

Md Salim Reza

Student ID: 012203028

A Project

in

The Department

of

Computer Science and Engineering

Presented in Partial Fulfillment of the Requirements

For the Degree of Master of Science in Computer Science and Engineering

United International University

Dhaka, Bangladesh

January 2024

Md. Salim Reza

Approval Certificate

This project titled "**Deep RL Conversational Chatbot**" submitted by [**Md. Salim Reza**], Student ID: [**012203028**], has been accepted as Satisfactory in fulfillment of the requirement for the degree of Master of Science in Computer Science and Engineering on [02-01-2022].

Board of Examiners

1.

_____ Supervisor

Prof. Dr. Swakkhar Shatabda
Professor & Director - B.Sc. in Data Science
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

2.

_____

Dr. Md. Saddam Hossain Mukta Examiner & Reviewer
Associate Professor & Program Coordinator - BSCSE
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

3.

_____ Ex-Officio

Md. Motaharul Islam, PhD
Professor & Director - MSCSE
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Declaration

This is to certify that the work entitled "**Deep RL Conversational Chatbot**" is the outcome of the research carried out by me under the supervision of **Dr. Swakkhar Shatabda, Professor, CSE and Director -MSCSE, UIU**.

_____

Md. Salim Reza
Student ID: 012203028
MSCSE Program
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

In my capacity as supervisor of the candidate's project, I certify that the above statements are true to the best of my knowledge.

_____

Prof Dr. Swakkhar Shatabda
Professor & Director - B.Sc. in Data Science
Department of Computer Science and Engineering (CSE)
United International University (UIU)
Dhaka-1212, Bangladesh

Abstract

Machine translation in natural language has made remarkable strides, particularly in language translation. Despite extensive global research, the generation of dialogue remains challenging, and this area continues to captivate researchers. Current dialogue generation models often yield inconsistent and generic answers, resulting in less engaging user conversations. This study explores recent methodological proposals by NLP researchers, focusing on reinforcement learning principles.

A novel reinforcement learning model for dialogue generation shows promise in generating conversational agent responses. However, its short memory can impact future results and predict utterances without considering the overall dialogue direction, affecting coherence and interest. An updated seq2seq model, employing reinforcement learning with the policy gradient method, encourages agents to participate in more engaging dialogues. The impact of reward functions, such as semantic coherence, controls conversation quality, and model evaluation includes quantitative measurements like n-gram iterations.

The goal of this research is to develop an open-domain AI conversational chatbot using reinforcement learning. By applying recent research models, the aim is to find an optimal hybrid solution for generating more interesting system utterances. Chatbots, integral in virtual assistants and messaging apps, are growing in popularity, making this project timely and relevant.

The system comprises three models: the Natural Language Understanding (NLU) module, the Dialogue Manager (DM) module, and the Reinforcement Learning algorithm. The NLU module analyzes user utterances, employing an ensemble machine learning strategy with bi-directional LSTM for intent classification and entity extraction. The NLU output serves as input for the DM module, where reinforcement learning guides decision-making to maximize rewards based on current and past status. The DM module aims to achieve human-level learning performance swiftly and facilitate engaging, meaningful, and logical discussions.

Acknowledgement

I wish to convey my heartfelt gratitude to the Almighty **Allah** for granting me the strength and perseverance to successfully complete this project.
Next, I extend my sincere appreciation and deepest thanks to my supervisor, **Prof Dr. Swakkhar Shatabda**, for his unwavering guidance, support, and invaluable insights throughout the entire research journey. His expertise and encouragement have played a pivotal role in shaping the direction and quality of this work.
I would also like to express my profound thanks to the project examiner, **Dr. Md. Saddam Hossain Mukta**, whose insights and teachings have enriched my understanding of various interesting topics, significantly contributing to the successful completion of this challenging project.
My gratitude knows no bounds, and I pray that Allah blesses each one of you for your contributions to this endeavor.

Table of Contents

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Introduction

Machine translation of natural language has done an impressive job in language translation. With so much research being done around the world, the generation of dialogue is still impractical, and so far, this topic has been a fascinating area of active research. The answers learned were inconsistent and general, and some conversations were not interesting, so the user's conversation was not very appealing. Here we examine the methodological proposals recently published by NLP researchers to draw the principles of reinforcement learning. The new reinforcement-learning model for dialog generation provides a great perspective for generating conversation agent responses, but the short memory of the model ignores its impact on future results and predicts utterances at some point. Tend to be modeling the future direction of the dialogue is important to make the dialogue coherent and interesting.

Updated seq2seq model using the reinforcement learning, policy gradient method encourages agents to engage in interesting dialogues[1] . Impact of reward functions such Click or tap here to enter text.as semantic coherence during simulated inter-agent conversations, semantic Coherence, Information flow and Ease of answering control of conversation quality, and model evaluation on quantitative measurements of language diversity such as n-gram iterations [3]. The modular dialog system for completing tasks has the major drawback of requiring each module to be trained individually. The problem with this model is that the previous module [4] affects the downstream module. To converse the model performance with humans, The very crucial part is policy optimizations which attempts to find a policy describing how to respond to humans. To solve this problem. particular attention is given to actor-critic methods, off-policy reinforcement learning with experience replay, and various methods aimed at reducing the bias and variance of estimators [2].

1.2 Motivation

The motivation behind why I am diving into the world of dialogue generation and reinforcement learning? Well, it all boils down to a big challenge – making conversations with machines more interesting and natural. Right now, when we talk to chatbots or virtual assistants, the responses can be a bit dull and predictable.
The goal here is to amp up the game. I expect chatbots to be not just smart but also good at keeping us engaged in conversations. The existing models have their limitations, especially in predicting where a conversation should go next.
Enter reinforcement learning – a cool concept where machines learn from rewards. I want to use this idea to teach our chatbots to give responses that people find more enjoyable. It is like training them to be better conversational partners.
This journey is motivated by the desire for chatbots that not only understand what we say but also respond in a way that feels natural and interesting. In this project work I explored different tricks and techniques, like tweaking how we measure success and using modular systems, to see how we can make our chatbots the best conversation buddies out there.

In a nutshell, my motivation is to make chatting with machines a lot more fun and engaging!

1.3 Project Goals

The overarching goal of this research project is to create and develop an open-domain AI conversational chatbot, placing a specific emphasis on integrating reinforcement learning (RL) within the dialogue manager to enhance the system's capacity for generating engaging responses. Drawing inspiration from recent advancements in research models, the objective is to craft an optimal hybrid solution that leverages RL techniques alongside other pertinent methodologies. This project seeks to contribute to the evolving landscape of chatbot technology, with applications in virtual assistants like Amazon's Alexa and Google Assistant, as well as messaging platforms such as WeChat. Through the implementation of RL in the dialogue manager, the aim is to enable the chatbot to dynamically generate responses based on a reward-driven policy, thereby elevating the quality and relevance of interactions. In addition to advancing the field of conversational AI, this research project aims to underscore the author's proficiency in integrating cutting-edge technologies, thereby fostering a deeper understanding of AI-driven dialogue systems.

1.4 Possible Applications

The envisioned open-domain AI conversational chatbot, driven by a reinforcement-learning model, offers a wide range of applications. From transforming customer support systems to enhancing e-learning experiences, acting as a virtual assistant for personal productivity, and facilitating language learning, the chatbot's versatility extends to healthcare, travel planning, and entertainment. In professional settings, it serves as a business intelligence tool and aids human resources. Innovative applications include interactive storytelling and providing empathetic responses in mental health support.

1.5 Software Architecture

The proposed software architecture for the entire system is illustrated in Figure 1.1. The system is intricately divided into three distinct models to optimize its functionality. The Natural Language Understanding (NLU) module takes charge of evaluating user utterances, employing a (Seq2Seq) Encoder and Decoder to effectively complete this task. The output vector generated by the Encoder serves as input to the Decoder, ensuring a seamless flow of information.

In the Dialogue Manager (DM), a Policy Gradient Reinforcement Learning (RL) model is implemented. This model is designed to generate the best policy for actions based on the current and previous states. The DM, equipped with this policy, strategically selects actions to maximize rewards, aiming for optimal performance. One of the primary objectives of the DM module is to achieve a level of learning performance comparable to human capabilities swiftly. Moreover, it strives to establish engaging, meaningful, and logically coherent conversations. The integration of these modules within the overall software architecture forms a robust framework for the development of the open-domain AI conversational chatbot.

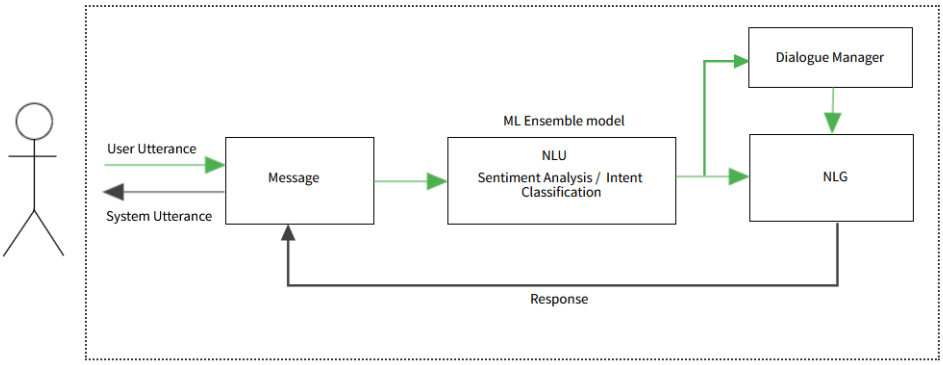

Figure 1. 1: Software Architecture

1.6 Organization of the Report

In this section, I provide a concise overview of the entire report.

Chapter 2 explores the **Background and Literature Review**, offering insights into the various RL and Deep Learning algorithms utilized in constructing the project model. This section also comprehensively reviews prior works conducted in different years, establishing a contextual foundation.

Chapter 3 details the **methodology, results and outputs** of the entire project, providing a comprehensive examination of Data Analysis, Data Preprocessing, Model Building, and Performance Measurement. Each step's intricacies are elucidated, offering a clear understanding of the procedural aspects of the project. Readers gain insights into the performance and accuracy of each algorithm employed in the project, facilitating the identification of the most effective algorithm.

Finally, **Chapter 4** serves as the **conclusion, Limitations and Future Works** of this project, encapsulating key findings and concluding remarks.

Chapter 2

Literature Review and Background

2.1 Literature Review

Machine translation of natural language has achieved remarkable success in language translation; however, the challenge of generating coherent and engaging dialogues remains an active area of research [5]. Despite extensive global efforts, current dialogue generation often produces inconsistent and general responses, leading to uninteresting conversations for users. Recent proposals by NLP researchers focus on reinforcement learning principles to address this challenge [5].

The introduction of a novel reinforcement-learning model for dialogue generation offers a promising perspective for enhancing conversation agent responses [2]. However, the model's short memory may impact future outcomes, predicting utterances at a specific point and potentially compromising coherence. Recognizing the importance of modeling the future direction of dialogues, researchers aim to ensure dialogues are not only coherent but also captivating.

An updated sequence-to-sequence (seq2seq) model, leveraging reinforcement learning through the policy gradient method, encourages agents to participate in more engaging dialogues [1]. Research emphasizes the impact of reward functions, including semantic coherence, during simulated inter-agent conversations [3]. Factors such as semantic coherence, information flow, ease of answering, and control of conversation quality play crucial roles, with model evaluation extending to quantitative measurements of language diversity, such as n-gram iterations [3].

While the modular dialog system for task completion offers versatility, it requires individual training for each module, posing a drawback as downstream modules may be influenced by previous ones [4]. To evaluate model performance against human interactions, policy optimization becomes a critical aspect [2]. Attention is directed toward actor-critic methods, off-policy reinforcement learning with experience replay, and various techniques aimed at reducing bias and variance in estimators [2].

The overarching goal of this research is to create and develop an open-domain AI conversational chatbot [5]. The focus is on generating more interesting system utterances through the implementation of a reinforcement-learning model [5]. By leveraging recently proposed research models [5], the objective is to find an optimal hybrid solution [5]. Chatbot technology, gaining popularity in virtual assistants like Amazon's Alexa, Google Assistant, and messaging apps such as WeChat [5], underscores the project's purpose—to create a chatbot utilizing reinforcement learning in the dialogue manager to generate optimal responses based on reward and policy [5]. This research aims to contribute to the evolving field of conversational AI [5], offering a comprehensive exploration of methodologies and models to enhance the quality and engagement of chatbot interactions. In Table 2.1 You can see a summary of Literature review.

Table 2. 1: Summary of Literature Review

Paper	Author	Contributions
Deep Reinforcement Learning for Dialogue Generation (2016)	Jiwei Li et. Al	Modeling future direction, minimize short sighted. Calculate future reward by RL policy gradient. Build a model on the long term success of dialogues., Heuristic approximation to rewards.
Sample efficient deep reinforcement learning for dialogue systems with large action spaces (2018)	Gellert Weisz et. Al	Policy optimization, in the form of a function taking current state of the dialogue and returning the response of the system. Given the attention to actor-critic methods, use off-policy RL, bias and variance reducer estimator. They studied at observable env. and relatively small set of actions.
Dialogue Generation using Reinforcement Learning and Neural Language Models (2016)	Marcella et. Al	Seq2Sec Conversational agent, Policy Gradient , Effects of Reward (SC,IF,EA)
End-to-End Task-Completion Neural Dialogue Systems (2018)	Yun-Nung Chen et. al.	Overcome of modularized task completion problem - downstream module affected by earlier module (both module need training). Proposed a novel end-to-end learning framework. End-to-End model learning mapping between DH to CR and train whole system.
Self-improving Chatbots based on Reinforcement Learning (2019)	Elena Ricciardelli	Policy learning is implemented using a Deep Q-Network (DQN) agent with epsilon-greedy exploration. Log all conversation at developing and testing phase with user feedback as supervised dataset feedback as the target, Then developed a score model for predicting user feedback at live stage. These predicted feedback then used as rewards for DQN RL.

The current landscape of dialogue generation in natural language processing has seen significant advancements, particularly in machine translation for language translation purposes [5]. However, a notable gap exists in the domain of generating coherent and engaging dialogues. While machine translation has made strides, dialogue generation still

faces challenges in producing responses that are consistently coherent and interesting for users. Existing dialogue generation models often yield inconsistent and generic responses, resulting in less engaging and meaningful conversations [5]. To address this gap, recent research endeavors have focused on applying reinforcement learning principles to enhance dialogue agent responses [5]. While these efforts show promise, they raise concerns about the model's short-term memory and its potential impact on maintaining conversation coherence. Additionally, there is a need to further explore how to ensure that dialogues not only exhibit semantic coherence but also captivate users. This presents an opportunity for the development of novel dialogue generation models that strike a balance between reinforcement learning and maintaining coherent, captivating conversations [2]. Moreover, the evaluation of such models should extend beyond traditional metrics to encompass a holistic assessment of language diversity and other critical factors [3]. The gap analysis reveals a promising direction for future research in dialogue generation, focusing on refining reinforcement-learning-based models to create more engaging and coherent conversational agents, ultimately contributing to the advancement of conversational AI technology [5].

2.2 Background Study

I delving into the challenge of making machine talk like humans. While translating languages is impressive progress, creating interesting and natural machine conversations remains difficult. In Natural Language Processing (NLP), researchers use reinforcement learning to improve how machine respond when we talk to them. My approach involves a sequence-to-sequence and RL model, a setup that helps machine engage in more interesting conversations over time. Researchers focus on rewards, ensuring the machine's responses make sense and are diverse.
The challenge is training each part individually, and what one part learns can affect others. Using policy optimization, My aim to make machine's responses comparable to human conversations. My goal is to create a chatbot that can chat about anything, blending the latest research ideas for optimal results. In the exciting world of conversational AI, I am on a journey to make this happen.

2.2.1 Current State of Art:

The current state of conversational AI is defined by notable progress in natural language processing, machine learning, and deep learning. Advanced models, often utilizing transformer architectures like GPT-3, exhibit exceptional capabilities in understanding and generating human-like responses. These large pre-trained language models excel in diverse language tasks, leveraging extensive data and sophisticated training methods. Transfer learning is a key strategy, enabling models pre-trained on large datasets to adapt and excel in various conversational domains, from customer support to virtual assistants. Reinforcement learning plays a crucial role in improving conversational agents, training models to optimize responses for more meaningful and context-aware interactions. Ongoing research addresses challenges like bias and ethical considerations, focusing on generating coherent and contextually relevant responses. Evaluation metrics for conversational AI emphasize factors such as relevance, coherence, and user satisfaction. The current state-of-the-

art highlights powerful transformer-based models, transfer learning, and a growing emphasis on reinforcement learning, pushing towards more natural and effective human-machine interactions.

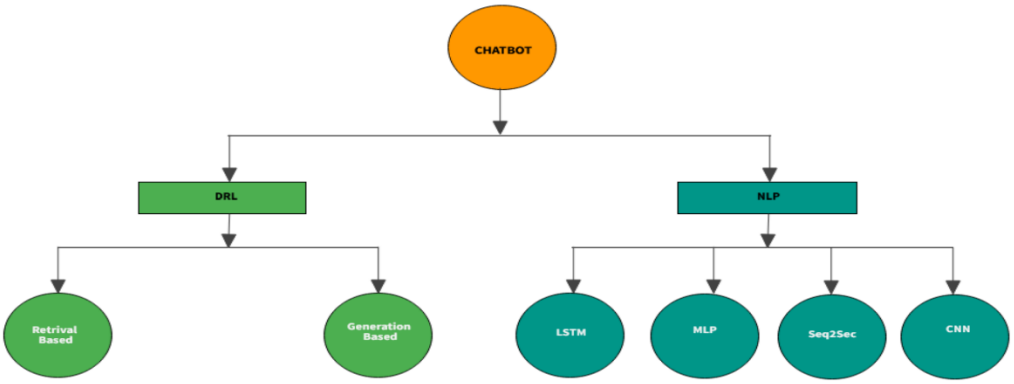

Figure 2. 1: Chatbot current State of Art

2.2.2 Seq2Seq Model

My chatbot's thinking process relies on a sequence-to-sequence (seq2seq) model, specifically designed to handle sequences of varying lengths as input and produce sequences of variable lengths as output, all within a fixed-sized model.

To accomplish this, Sutskever et al. proposed the use of two separate recurrent neural networks (RNNs). The first RNN serves as an encoder, transforming a variable-length input sequence into a fixed-length context vector. Essentially, this context vector, situated in the final hidden layer of the RNN, contains semantic information related to the input query sentence. The second RNN acts as a decoder. It takes an input word along with the context vector and predicts the next word in the sequence. Additionally, it generates a hidden state for the next iteration, allowing for an iterative process in sequence generation.

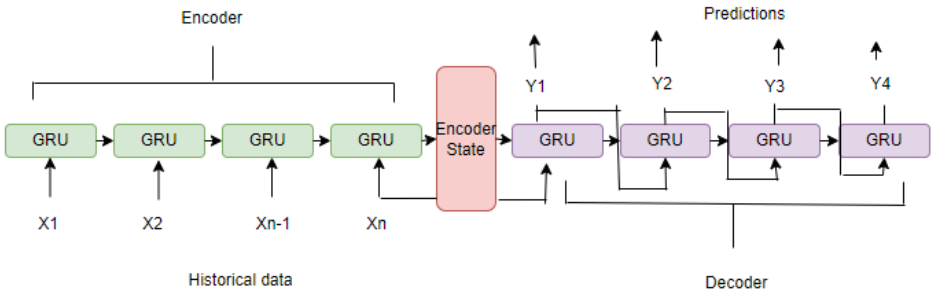

Figure 2. 2: Seq2Seq model

2.2.3 Encoder

An RNN (Recurrent Neural Network) encoder is a component of a neural network architecture designed to process sequential data, such as sentences or time-series data. In the context of natural language processing, an RNN encoder is specifically used to transform variable-length input sequences into a fixed-size representation, also known as a context vector.

Here is how an RNN encoder generally works:

1. **Sequential Processing:** The encoder processes the input sequence (e.g., a sentence) one token (word or character) at a time. It iterates through the sequence, updating its internal state at each step.
2. **Hidden States:** At each time step, the encoder produces both an "output" vector and a "hidden state" vector. The hidden state contains information about the input sequence seen so far. This hidden state is passed to the next time step, allowing the encoder to maintain a memory of the entire input sequence.
3. **Context Vector:** The final hidden state, often considered as a context vector, encapsulates the summarized information from the entire input sequence. This context vector is used to represent the input sequence in a fixed-size form.

The purpose of the RNN encoder is to capture the sequential dependencies and contextual information in the input data, condensing it into a form that retains essential information. This fixed-size representation can then be passed to other components of the neural network, such as a decoder in a sequence-to-sequence model.
It is worth noting that traditional RNNs suffer from some limitations, such as difficulty in capturing long-term dependencies. More advanced variants, like the Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM), have been introduced to address these challenges and are commonly used in modern encoder architectures.

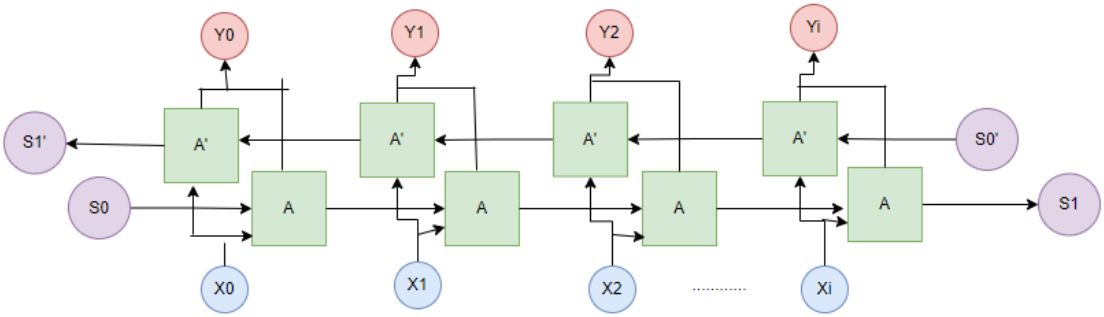

Figure 2. 3: Encoder Architecture

2.2.4 Decoder

In the context of neural networks, particularly in sequence-to-sequence models, a decoder is a component responsible for generating an output sequence based on the information provided by an encoder. It works in tandem with the encoder to process sequential input and produce sequential output.

Here is how a decoder generally functions:

1. **Context Initialization:** The decoder typically starts with an initial context or hidden state. This context is often derived from the final hidden state of the encoder, representing the summarized information about the input sequence.
2. **Sequential Generation:** Similar to the encoder, the decoder operates in a sequential manner. It processes the input sequence one element at a time, updating its internal state at each step.
3. **Output Generation:** At each time step, the decoder generates an output based on its current state and the context information. This output can be a word, token, or any relevant unit of the output sequence.
4. **Hidden State Update:** The decoder's hidden state is updated at each time step, incorporating information about the input sequence and the generated output so far.
5. **Iterative Process:** The decoder repeats this process until it produces the entire output sequence or reaches a predefined length.

In the context of natural language processing, the input sequence is often a sentence in one language, and the output sequence is the corresponding translation in another language. In other applications, the input-output pairs can represent a wide range of sequential data transformations. The decoder architecture is shown in Figure 2. 4.

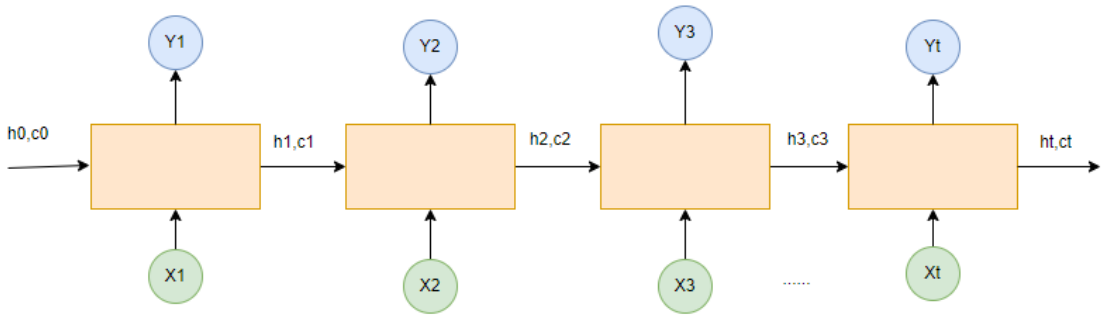

Figure 2. 4: Decoder Architecture

Chapter 3

Methodology

3.1 Data Set

DailyDialog is a superior multi-turn open-domain English dialogue dataset. The language is human-written, ensuring minimal noise. The dataset captures the essence of our daily communication, encompassing diverse topics related to everyday life. It consists of 13,118 dialogues, segmented into a training set with 11,118 dialogues, and validation and test sets, each comprising 1000 dialogues. On average, each dialogue involves approximately 8 speaker turns, with each turn containing around 15 tokens.

Table 3.1: Dataset Description

Description	Total
Total Dialogues	13,118
Training Set	11,118
Validation Set	1000
Test Set	1000

Source: https://paperswithcode.com/dataset/dailydialog

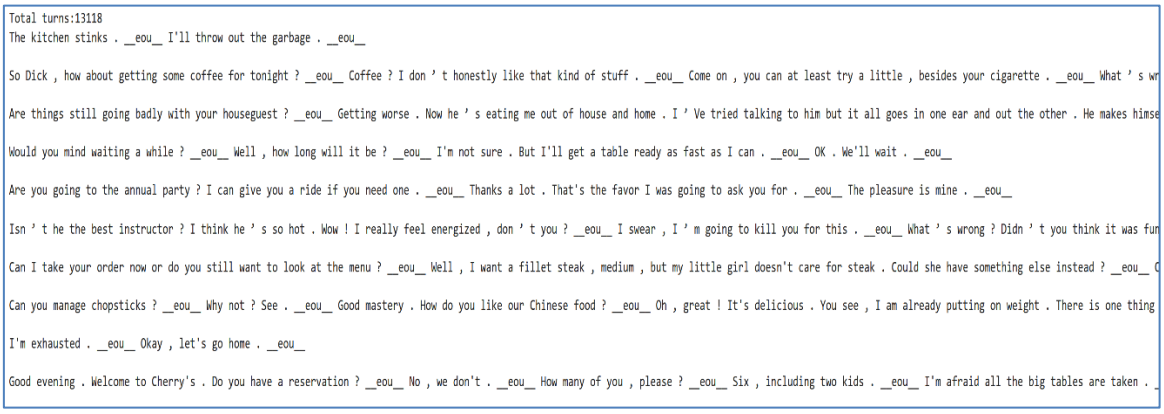

Figure 3. 1: Raw data (DailyDailog)

3.2 Data Processing and Preprocessing

3.2.1 Format Data:

To streamline the process, I generated a formatted data file where each line consists of a tab-separated pair, comprising a query sentence and its corresponding response. The formatting was facilitated by a custom data function called **splitConversation**. This Python function reads a conversation data file, initially breaking it into individual lines, and subsequently segmenting each line into distinct segments using the "eou" delimiter. In essence, **splitConversation** enhances the organization of the

data. Additionally, the **getDialogs** function is employed to extract pairs of sentences from these conversations.

Separate questions, answer, and used default word token to create word index

PAD_token = 0 - Used for padding short sentences
SOS_token = 1 - Start-of-sentence token
EOS_token = 2 - End-of-sentence token

In the data preprocessing phase, several crucial steps were undertaken to refine and optimize the input dataset. These steps include reducing the vocabulary by excluding rarely encountered words, addressing dull responses, and converting accented characters to their closest ASCII equivalents. Furthermore, a meticulous punctuation handling approach was implemented, involving the insertion of a space before each period, exclamation mark, and question mark. To enhance data cleanliness, consecutive spaces were removed.

Preserving the integrity of input sequences, a strategy was employed to ensure the retention of the last word for the End of Sentence (EOS) token. The culmination of these preprocessing steps is encapsulated in the functions defined throughout this process. The final output comprises a populated vocabulary object (voc) and a list of pairs, signifying a comprehensive and refined dataset ready for subsequent stages in the project. Figure 3. 2 represents the formatted data.

```
Preparing training data query/response..
Read 89862 sentence pairs
Trimmed to 43724 sentence pairs
Total words: 10126

pairs:
['good evening . welcome to cherry s . do you have a reservation ?', 'no we don t .']
['no we don t .', 'how many of you please ?']
['how many of you please ?', 'six including two kids .']
['six including two kids .', 'i m afraid all the big tables are taken .']
['what kind of food do you like ?', 'i like chinese food .']
['i like chinese food .', 'but your american ?']
['but your american ?', 'we have a lot of chinese restaurants in america .']
['would you like to take a look at the menu sir ?', 'yes . thank you .']
['yes . thank you .', 'would you care for a drink before you order ?']
['would you care for a drink before you order ?', 'a glass of qingdao beer .']
```

Figure 3. 2: Formatted data

3.3 Data Preprocessing for Model

In preparing the data for the model, a key consideration is accommodating sentences of different lengths within a single batch. This is achieved by creating a batched input tensor with dimensions (max_length, batch_size), where shorter sentences are zero-padded after the End of Sentence (EOS) token.

The **'getInput'** function is instrumental in converting sentences into tensors, producing a tensor of lengths for each sequence in the decoder's batch. Similarly, the **'getOutput'**

function, mirroring the functionality of 'getInput', generates a binary mask tensor and determines the maximum target sentence length.

The **'getTrainDataFromBatch'** function plays a central role in processing pairs, producing input and target tensors. This comprehensive data processing approach ensures that the model can effectively handle varying sentence lengths and is well-prepared for training with diverse input sequences. Figure 3. 3 shows the tensor data.

Input tensor is a tensor containing the padded input sequence along with their corresponding lengths

```
input_variable: tensor([[ 489,   80,   96,  117,  176],
        [   8,   41,  310,  183,   65],
        [ 278,  107,  941,   15,   48],
        [ 124,   57,   15,   74, 4845],
        [ 552,   62,   41,    8,   30],
        [   8,    8,   16,   28,    2],
        [  41,   42,  138,  853,    0],
        [ 195,   35,    8,  232,    0],
        [  45,  432,   30,   15,    0],
        [3392,   15,    2,    2,    0],
        [ 627,    2,    0,    0,    0],
        [  15,    0,    0,    0,    0],
        [   2,    0,    0,    0,    0]])
```

Figure 3. 3: Tensor data

Output and Mask tensor: preparing a padded target sequence tensor and corresponding padding mask for a given set of sentences. It first transforms the input sentences into sequences of word indexes, subsequently determining the maximum length among these sequences. The sequences are then zero-padded to align with the maximum length, and a binary mask is created to identify the padding positions. The result is a set of tensors conducive to the model's training process, ensuring compatibility with varying sentence lengths. Figure 3. 4 represents the mask tensor.

```
target_variable: tensor([[  54,   35,   96,   74, 3767]
        [  16,   65,  310,    8,   15],
        [  12,    5,   15,   28,    2],
        [  23,   45,   16,   48,    0],
        [ 374,  143,   60, 2478,    0],
        [  56,   30, 1619,   84,    0],
        [  35,    2,   28,   76,    0],
        [ 494,    0,   45,   15,    0],
        [  30,    0,  316,    2,    0],
        [   2,    0, 1260,    0,    0],
        [   0,    0,   15,    0,    0],
        [   0,    0,    2,    0,    0]])
```

```
mask: tensor([[ True,  True,  True,  True,  True],
        [ True,  True,  True,  True,  True],
        [ True,  True,  True,  True,  True],
        [ True,  True,  True,  True, False],
        [ True,  True,  True,  True, False],
        [ True,  True,  True,  True, False],
        [ True,  True,  True,  True, False],
        [ True, False,  True,  True, False],
        [ True, False,  True,  True, False],
        [ True, False,  True, False, False],
        [False, False,  True, False, False],
        [False, False,  True, False, False]])
```

Figure 3. 4: Mask Tensor

3.4 Seq2Seq model

The chatbot's brain is a sequence-to-sequence model. One functions as an encoder, converting a variable-length input sequence into a fixed-length context vector. The second as a decoder, using a word and a context vector as input to predict the next word in the sequence.

3.4.1 Encoder:

The encoder RNN plays a crucial role in processing tokens, generating "output" and "hidden state" vectors at each step. The hidden state is sequentially propagated, while the output vectors are preserved for further use. This encoder employs a bidirectional multi-layered Gated Recurrent Unit (GRU), allowing it to capture both past and future contextual information. The computational graph unfolds as follows:

1. **Word Embedding:** Convert word indices to embeddings.
2. **Padded Sequence Preparation:** Organize the sequence batch with appropriate padding for the RNN module.
3. **Forward Pass through GRU:** Execute the forward pass through the bidirectional GRU.
4. **Padding Removal:** Eliminate padding from the GRU outputs.
5. **Combine Bidirectional Outputs:** Integrate the outputs from the bidirectional GRU to capture a holistic representation.
6. **Output and Hidden State Provision:** Supply the final output and hidden state for subsequent model processing.

3.4.2 Decoder:

The decoder RNN is instrumental in generating the next word in the sequence, leveraging both the context vectors from the encoder and its internal hidden states. This iterative process continues until an End of Sentence (EOS_token) is encountered. However, a drawback of the conventional seq2seq decoder becomes apparent when relying solely on the context vector, leading to information loss, especially with longer input sequences.

To mitigate this issue, the integration of an attention mechanism proves invaluable. This mechanism empowers the decoder to selectively concentrate its attention on specific segments of the input sequence, thereby enhancing its ability to handle and capture intricate information from lengthier input sequences.

The Luong Attention Model employs a detailed computation graph to illustrate its functionality in sequence-to-sequence learning. This model is particularly adept at leveraging attention mechanisms to enhance the model's capacity for capturing pertinent information from input sequences during decoding. The key steps in this computation graph include:

1. **Embedding Step:** Obtain the embedding of the current input word.
2. **GRU Forward Pass:** Pass the embedded word through a unidirectional Gated Recurrent Unit (GRU) to generate an intermediate output.

3. **Attention Weights Calculation:** Compute attention weights based on the current GRU output and the encoder outputs.
4. **Context Vector Generation:** Multiply the attention weights with the encoder outputs to produce a new context vector.
5. **Luong Concatenation:** Concatenate the weighted context vector and the GRU output using the Luong equation.
6. **Next Word Prediction:** Predict the next word in the sequence using the concatenated vector.
7. **Output and Hidden State:** Provide the predicted output along with the final hidden state of the GRU.

This detailed process delineates the intricate interplay between attention mechanisms and the decoding process, showcasing the Luong Attention Model's effectiveness in capturing and utilizing relevant information for improved sequence generation.

3.5 Training

The task progression involves the following steps:

1. **Forward Pass through Encoder:** Process the complete input batch through the encoder in a forward pass.
2. **Decoder Initialization:** Begin the decoder inputs with the Start of Sentence (SOS_token) and set the initial hidden state as the final hidden state of the encoder.
3. **Decoder Forward Pass:** Pass the input batch sequence through the decoder.
4. **Update Decoder Input:** Set the subsequent decoder input as the ongoing target; otherwise, assign the subsequent decoder input as the ongoing decoder output.
5. **Loss Computation:** Compute and gather the loss.
6. **Backpropagation:** Conduct backpropagation.
7. **Gradient Clipping:** Apply gradient clipping.
8. **Model Revision:** Revise the encoder and decoder.

The function `maskNNLLLoss` is responsible for calculating the average negative log likelihood of elements in a tensor based on a mask tensor. It specifically focuses on elements in the input tensor corresponding to a value of 1 in the mask tensor. The function returns both the computed loss and the total number of non-masked elements, providing valuable metrics for evaluating the model's performance.

In the implementation of PyTorch's RNN modules, such as RNN, LSTM, and GRU, they can be seamlessly integrated into the model like any other non-recurrent layers. Despite the abstraction, it is crucial to recognize the iterative process happening under the hood, where hidden states are computed at each time step. In our encoder, we leverage the GRU layer for the sequential processing of input sequences or batches. While these modules can process entire sequences at once, during training, we manually loop over sequences, akin to what is done for the decoder model. Maintaining the correct conceptual model of these modules simplifies the implementation of sequential models.

The training procedure is consolidated in the **trainIters** function, responsible for executing n_iterations of training with the provided models, optimizers, and data. This function, building upon the train function, streamlines the training process. A notable aspect is the

model-saving mechanism, which creates a tarball containing the state_dicts of the encoder and decoder (parameters), optimizers' state_dicts, loss, iteration, and more. This approach ensures maximum flexibility when working with checkpoints. Loading a checkpoint allows us to use the model parameters for inference or seamlessly resume training from where we left off.

I used a module is specifically crafted to facilitate greedy decoding within sequence-to-sequence models. In the sequence generation, greedy decoding involves choosing the token with the highest probability at each step of the process.

3.6 Evaluation of model

The evaluation of the model involves a sequence-to-sequence architecture for natural language processing. In this design, the encoder processes the input sentence, while the decoder is responsible for generating an output sequence. Throughout the decoding phase, a searcher comes into play, tasked with selecting tokens according to a designated strategy, such as the widely employed greedy decoding approach. This evaluation framework aims to assess the model's proficiency in accurately transforming input sentences into coherent and contextually relevant output sequences.

3.7 Reinforcement Learning (RL)

In this section, a comprehensive exploration of the components constituting the Reinforcement Learning (RL) mode. The learning system is characterized by the involvement of two agents denoted as the first agent generating sentences (p) and the second agent generating sentences (q). The dialogue unfolds as an alternating sequence of sentences from these agents: p_1,q_1 ,p_2 , q_2, ..., p_i, q_i. Each sentence generated is considered as an action dictated by a policy established through an encoder-decoder recurrent neural network language model.

The network's parameters undergo optimization to maximize the expected future reward, employing policy search. Policy gradient methods are favored in this scenario over Q-learning due to the advantage of initializing the encoder-decoder RNN with Maximum Likelihood Estimation (MLE) parameters that already yield plausible responses. This initialization allows for a smoother transition towards tuning the objective to maximize long-term reward. In contrast, Q-learning directly estimates the future expected reward for each action, presenting challenges when MLE parameters, optimized for a different objective, are used for initialization.

The sequential decision problem at hand encompasses various components, including states, actions, rewards, etc., which are elaborated upon in the subsequent sub-sections.

3.7.1 Action

An action, denoted as "a," represents the dialogue utterance to be generated. The action space is infinite, given that arbitrary-length sequences have the potential to be generated.

3.7.2 State

A state is represented by the preceding two dialogue turns $[p_i, q_i]$. To create a vectorized representation of the dialogue history, the concatenation of pi and qi is input into an LSTM encoder model.

3.7.3 Policy

A policy is structured as an LSTM encoder-decoder, denoted as $pRL(p_i+1 \mid p_i, q_i)$, and is characterized by its parameters. It is important to highlight that we adopt a stochastic representation of the policy, meaning it is a probability distribution over actions given states. This choice is made to avoid a deterministic policy, as it would lead to a discontinuous objective that poses challenges for optimization using gradient-based methods.

3.7.4 Reward

In the context of reinforcement learning, a reward is a numerical value assigned to an agent's action in an environment, indicating the desirability or success of that action. The agent's objective is to learn a policy that maximizes the cumulative reward over time. The reward will be Positive if generate coherent, contextually relevant, and engaging responses. In this project, various types of rewards have been employed:

Ease of Answering: Encourages the machine-generated turn to be easy for a user to respond to. Promotes the creation of dialogue responses that facilitate continued and meaningful conversation.

Information Flow: Encourages the agent to contribute new information in each turn to maintain a dynamic and engaging dialogue. Aims to avoid repetitive sequences and ensures the conversation remains informative and interesting.

Semantic Coherence: Measures the adequacy of responses by assessing their grammatical correctness and coherence. Ensures that generated replies, while being rewarded, are also linguistically sound and contextually appropriate.

These reward criteria collectively contribute to the overall quality of the machine-generated dialogue by addressing aspects of user engagement, information richness, and linguistic coherence.

The final reward for action a is a weighted sum of the rewards discussed above:

$$R = \sum_{t=0}^{\infty} \gamma^t R_{t+1} = R_1 + \gamma R_2 + \gamma^2 R_3 + \ldots,$$

Here, R is the weighted sum of the discounted rewards. In the context of reinforcement learning, R_{t+1} represents the reward received when transitioning from state S_t ***to*** S_{t+1} The parameter $0 \leq \gamma < 1$ is the discount rate. A discount rate less than 1, denoted by γ, implies that rewards obtained in the distant future are given less weight compared to rewards obtained in the immediate future.

3.8 Simulation between Two Agents

In this project, simulate dialogues between two virtual agents, allowing them to engage in a conversational exchange. The simulation unfolds in the following manner: initially, a message from the training set is provided to the first agent. This agent then encodes the input message into a vector representation and commences the decoding process to produce a response. Incorporating the immediate output from the first agent alongside the dialogue history, the second agent adjusts its state by encoding the entire dialogue history into a representation. The second agent utilizes a decoder RNN to generate responses, which are subsequently looped back to the first agent. This iterative process continues as the virtual agents take turns conversing. Figure 3. 5 depicts, simulation between two virtual agents.

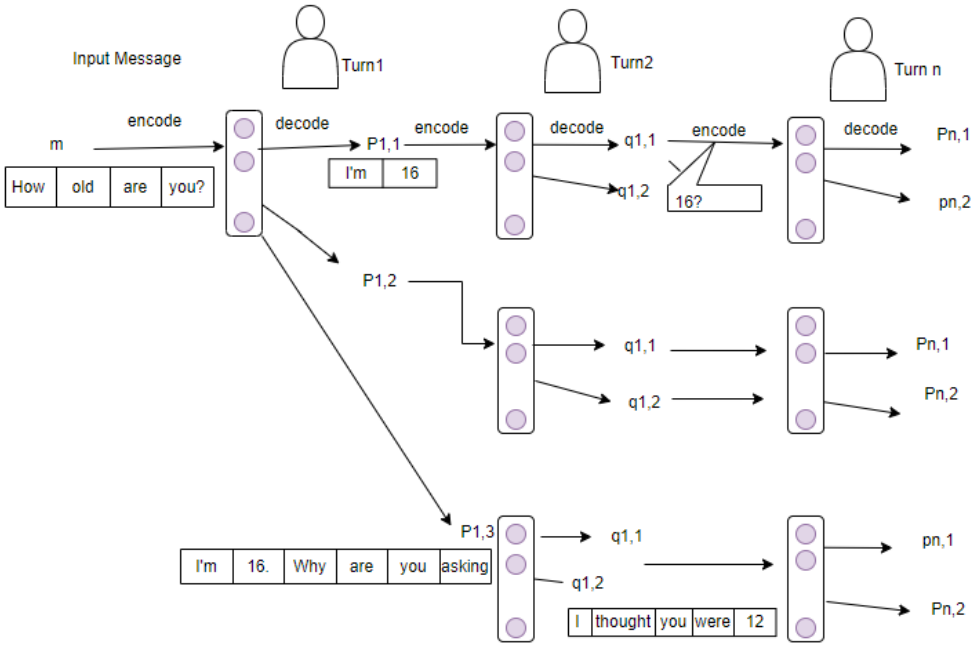

Figure 3. 5: Simulation by two virtual agents

3.9 Optimization

In this project, the policy model was initialized with parameters derived from the mutual information model. Leveraging policy gradient methods, the aim was to discover parameters that maximize the expected future reward. The primary objective was to enhance the expected reward resulting from specific actions. Employing the likelihood ratio trick facilitated efficient gradient updates during the optimization process.

3.10 Result

It is tough to evaluate dialogue systems. People usually use metrics like BLEU and perplexity to measure how good the dialogues are, but there's a lot of debate about how well these metrics really show if the responses are good. My project's main goal isn't just about predicting the most likely response; it's more about making sure the conversation stays successful for a long time. That's why not used metrics like BLEU and perplexity for evaluation. Instead, focused on a broader assessment that fits with the overall goal of the system. I was looking at the detailed dynamics and the things that make conversations successful over time. Figure 3.6 shows the chat sequence of RL Chatbot Response

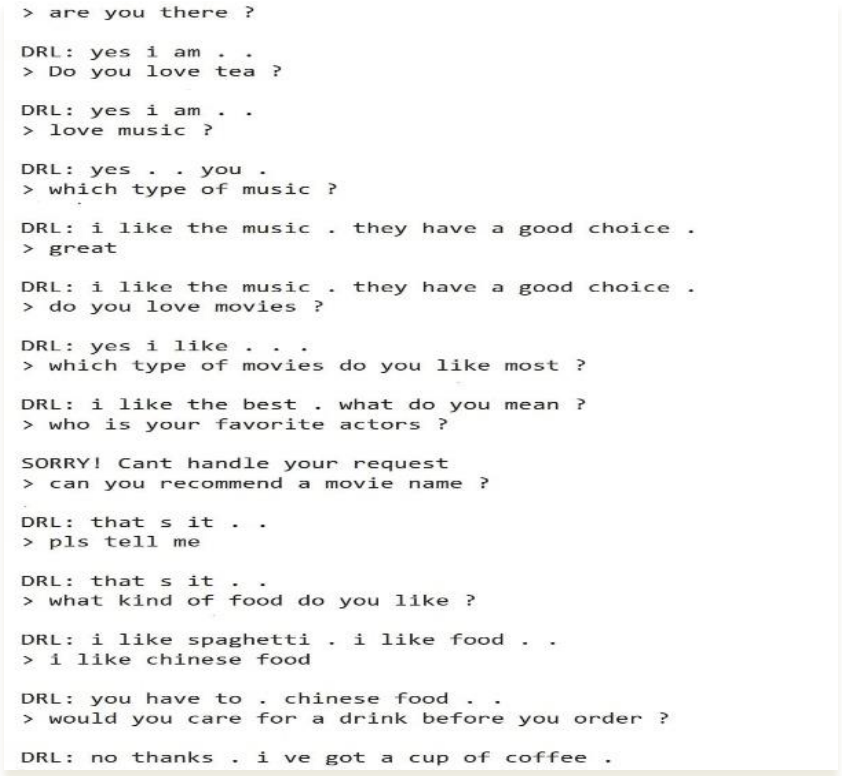

Figure 3. 6: RL Chatbot Response

Chapter 4
Conclusion and Limitations

4.1 Conclusion

In this project, I employed a reinforcement learning framework to enhance neural response generation by simulating dialogues between two agents. This approach combines the strengths of neural SEQ2SEQ systems and reinforcement learning, aiming to capture the compositional models of dialogue meaning and generate responses that align with the semantic context. Similar to earlier neural SEQ2SEQ models, the employed reinforcement learning framework excels in understanding dialogue turns and producing semantically appropriate responses. Furthermore, drawing inspiration from reinforcement learning dialogue systems, it effectively generates utterances that optimize future reward, thereby encapsulating global properties of a good conversation. Despite the simplicity of the heuristics used in the model to capture these global properties, the framework manages to generate more diverse and interactive responses, contributing to a more sustained and engaging conversation.

4.2 Limitations

In conclusion, this project, used an extensive NLP dataset and employing a deep learning model, initially encountered significant challenges related to managing adequate computational resources, particularly high-end GPUs, for processing vast tensors using TensorFlow. Recognizing the technological hurdles, I sought and obtained permission from my project supervisor to transition from TensorFlow to PyTorch as the preferred programming framework. This adjustment proved crucial in overcoming the computational challenges, facilitating smoother model development, and ultimately contributing to the successful implementation of the project. The experience which i have adopt from this project will help me a lot to solve problem using RL and deep learning applications.

4.3 Future Works

In future endeavors, the goal is to enhance the accuracy of the Chatbot by incorporating more advanced AI and reinforcement learning (RL) algorithms. Expanding the model's capabilities with sophisticated algorithms will contribute to a more intelligent and context-aware conversational agent. Additionally, integrating a Bangla language dataset into the training process will be pivotal for extending the Chatbot's linguistic capabilities and ensuring effective communication in the Bangla language.

References

[1] Li, J., Monroe, W., Shi, T., Jean, S., Ritter, A., and Jurafsky, D. (2016). Deep reinforcement learning for dialogue generation. In Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pp. 1192-1202.

[2] G. Weisz, P. Budzianowski, P.-H. Su, and M. Gašić, "Sample Efficient Deep Reinforcement Learning for Dialogue Systems with Large Action Spaces," Feb. 2018, [Online]. Available: http://arxiv.org/abs/1802.03753

[3] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, "Deep Reinforcement Learning for Dialogue Generation." [Online]. Available: https://en.wikipedia.org/wiki/Yes,_and...

[4] X. Li, Y.-N. Chen, L. Li, J. Gao, and A. Celikyilmaz, "End-to-End Task-Completion Neural Dialogue Systems," Mar. 2017, [Online]. Available: http://arxiv.org/abs/1703.01008

[5] E. Ricciardelli and D. Biswas, "Self-improving Chatbots based on Reinforcement Learning," 2019. [Online]. Available: https://www.researchgate.net/publication/333203489

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of deep bidirectional transformers for language understanding," arXiv preprint arXiv:1810.04805, 2018.

[7] P. Michel, G. Neubig, and M. Auli, "Large scale pretraining for neural machine translation," arXiv preprint arXiv:1808.00202, 2018.

[8] T. Zhao, M. Eskenazi, and D. Radev, "DialogWAE: Multimodal response generation with conditional Wasserstein auto-encoder," in *Proceedings of the* 2018 Conference on Empirical Methods in Natural Language Processing, 2018, pp. 3017–3026.

[9] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Generating informative responses with controlled sentence function," arXiv preprint arXiv:1810.07942, 2018.

[10] Y. Adiwardana, M. Luong, D. R. So, J. Hall, N. Fiedel, R. Thoppilan, Z. Yang, A. Kulshreshtha, G. Nemade, Y. Lu, et al., "ChatGPT: Large-scale generative pre-training for conversational response generation," arXiv preprint arXiv:2111.05507, 2021.

[11] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "Controllable abstractive dialogue summarization with sketch supervision," arXiv preprint arXiv:1911.03876, 2019.

[12] B. Wu, J. Wang, X. Li, S. Yan, W. Tan, and X. Cheng, "Reinforcement learning based multi-turn response generation with adaptive rewards," arXiv preprint arXiv:2001.09382, 2020.

[13] C. Wu, S. Zhang, Y. Wu, W. Wu, Z. Li, M. Zhou, and Z. Yan, "Learning symmetric collaborative dialogue agents with dynamic knowledge graph embeddings," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 5924–5933.

[14] P. Shaw, J. Uszkoreit, and A. Vaswani, "Discrete reasoning over dialogs," arXiv preprint arXiv:1910.00464, 2019.

[15] S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, "Personalizing dialogue agents: I have a dog, do you have pets too?" arXiv preprint arXiv:1801.07243, 2018.

[16] Y. N. Dauphin, A. Fan, M. Auli, and D. Grangier, "Neural approaches to conversational AI," in Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, 2019, pp. 3543–3555.