Dialogue generation using Deep Reinforcement Learning

A PROJECT REPORT

In partial fulfilment of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING (CSE)

Submitted by

Karra Raghu Ram

Registration No – 14010816

DEPARTMENT OF COMPUTER SCIENCE AND ENGG.

VEER SURENDRA SAI UNIVERSITY OF TECHNOLOGY, BURLA

SAMBALPUR, ODISHA, INDIA- 768018

November 2017

DEPARTMENT OF COMPUTER SCIENCE AND ENGG.

VEER SURENDRA SAI UNIVERSITY OF TECHNOLOGY, BURLA

SAMBALPUR, ODISHA, INDIA- 768018

BONAFIDE CERTIFICATE

This is to certify that this project report entitled **“Dialogue generation using Deep Reinforcement Learning”** is a bonafide work of **Karra Raghu Ram** for partial fulfilment of the degree of Bachelor of Technology in Computer Science and Engineering of Veer Surendra Sai University of Technology, Burla, Odisha, who carried out the project work under my supervision.

H.O.D **SUPERVISOR**

ACKNOWLEDGEMENT

I would like to express my hearty thankfulness to all those who helped me during my project work, especially to my mentor Dr. Manas Ranjan Kabat for his valuable suggestions, motivations and guidance**.** His expertise and guidance proved to be valuable during every stage of the project.

I wish to express my gratitude to the HOD and other faculty members of computer science department for teaching me and helping me whenever required. Thank you, all teachers, you have been a great source of inspiration.

Lastly, I am thankful to my parents, without their support and inspiration, this project would not have been possible.

Karra Raghu Ram

ABSTRACT

Conventional Neural Networks in the past promise producing responses for conversational agents, with a shortcoming that they are short-sighted and ignore their effect on the future responses. The need to generate coherent and meaningful dialogues being the main goal, a need which led traditional NLP models of dialogue to draw on reinforcement learning.

This report shows how Long Short-term memory networks can be used to produce compound arrangements with extended range structure, just by forecasting one data point at a time. This model considers the following features while generating the sequences informativity and coherence. The accuracy score of this model can’t be mathematically found since there is no specific reply to a conversation. Hence this model is evaluated using length, diversity as well as human judges showing that the algorithm generates more interactive responses than the previous generation Neural Network Models.

CONTENTS

1. INTRODUCTION

Recurrent neural networks are a class of neural networks that have fixed cycle to themselves which they can use to develop successive inputs. In the case of dialogue creation, a dissimilar variation of recurrent neural network is used. LSTM networks have all the properties of recurrent neural networks aided by some features that permit it to learn long term needs.

LSTM block is itself a RNN as it contains recurrent connections like connections in a convolutional neural network. The main component of an LSTM block are as follows:

1. Cell
2. Input Gate
3. Output Gate
4. Forget Gate

LSTM takes full advantage of the probability of producing a response given the previous dialogue. Two difficulties arise in the traditional SEQ2SEQ LSTM model in dialogue generation. First, they use the MLE (Maximum likelihood) objective function which produce highly common responses like “I don’t know.” Irrespective of the input provided. Another common problem is that the system gets stuck in an endless loop of boring responses.

Overcoming these issues an encoder-decoder architecture is used as the core of an LSTM network. This simulates the discussion between two virtual agents to discover the state of possible actions while learning to maximize the probable reward. Hence to measure good conversations the following heuristics are defined:

1. Forward looking
2. Interactive
3. Informative
4. Coherent

This model thus assimilates SEQ2SEQ systems to learn compositional semantic gist of words as well as reinforcement learning in optimizing for long-term goals.

1.1 Domain of work

Artificial Neural Networks: ANN are associated set of artificial neurons where each connection can transfer data between two neurons which in turn process the input given by the previous layer of input and generate certain prediction and classification/regression to either be fed to the next layer or as an output. ANN are not always applied in their pure form rather they have variants which are application specific and produce better results than a conventional ANN. Some of the variants are:

1. Convolutional Neural Networks
2. Long Short-term Memory Networks
3. Deep Predictive Coding Networks

Reinforcement Learning: RL is a part of Machine Learning inspired by behaviourist thinking, concerned with how software agents take actions in a situation to maximize the collective reward. This is commonly termed as MDP (Makarov Decision Process) as many reinforcement learning algorithms use dynamic programming techniques.

Recurrent Neural Networks: RNN is a class of artificial neural network where connections between units form a directed cycle. This allows it to exhibit dynamic temporal behaviour. Unlike feedforward neural networks, Recurrent neural networks are a class of neural networks which have a directed cycle to themselves which they can use to process sequential inputs. In the case of dialogue generation, a different variation of recurrent neural network is used.

Long Short-term Memory Networks: LSTM block is itself a RNN as it contains recurrent connections like connections in a convolutional neural network. LSTM maximizes the probability of generating a response given the previous dialogue. Two problems arise in the traditional SEQ2SEQ LSTM model in dialogue generation. First, they make use of the MLE (Maximum likelihood) objective function which generate highly generic responses like “I don’t know.” Regardless of the input provided. Another common problem is that the system gets stuck in an infinite loop of repetitive responses.

1.2 Definitions and terminologies related to Dialogue Generation:

Reinforcement Learning: RL is a part of Machine Learning stimulated by behaviourist thinking, concerned by how software agents yield actions in a situation to maximize the collective reward. This is commonly termed as MDP (Makarov Decision Process) as many reinforcement learning algorithms use dynamic programming techniques.

Artificial Neural Networks: ANN are a connected set of artificial neurons where each connection can transfer information between two neurons which in turn process the input provided by the previous layer of input and generate certain prediction and classification/regression to either be fed to the next layer or as an output. ANN are not always implemented in their pure form rather they have variants which are application specific and produce better results than a conventional ANN. Some of the variants are:

1. Convolutional Neural Networks
2. Long Short-term Memory Networks
3. Deep Predictive Coding Networks

Recurrent Neural Networks: RNN is a class of artificial neural network where connections between units form a directed cycle. This allows it to exhibit dynamic temporal behaviour. Unlike feedforward neural networks, Recurrent neural networks are a class of neural networks which have a directed cycle to themselves which they can use to process sequential inputs. In the case of dialogue generation, a different variation of recurrent neural network is used.

Long Short-term Memory Networks: LSTM block is itself a RNN as it contains recurrent connections like connections in a convolutional neural network. LSTM maximizes the probability of generating a response given the previous dialogue. Two problems arise in the traditional SEQ2SEQ LSTM model in dialogue generation. First, they make use of the MLE (Maximum likelihood) objective function which generate highly generic responses like “I don’t know.” Regardless of the input provided. Another common problem is that the system gets stuck in an infinite loop of repetitive responses.

1.3 Illustrations

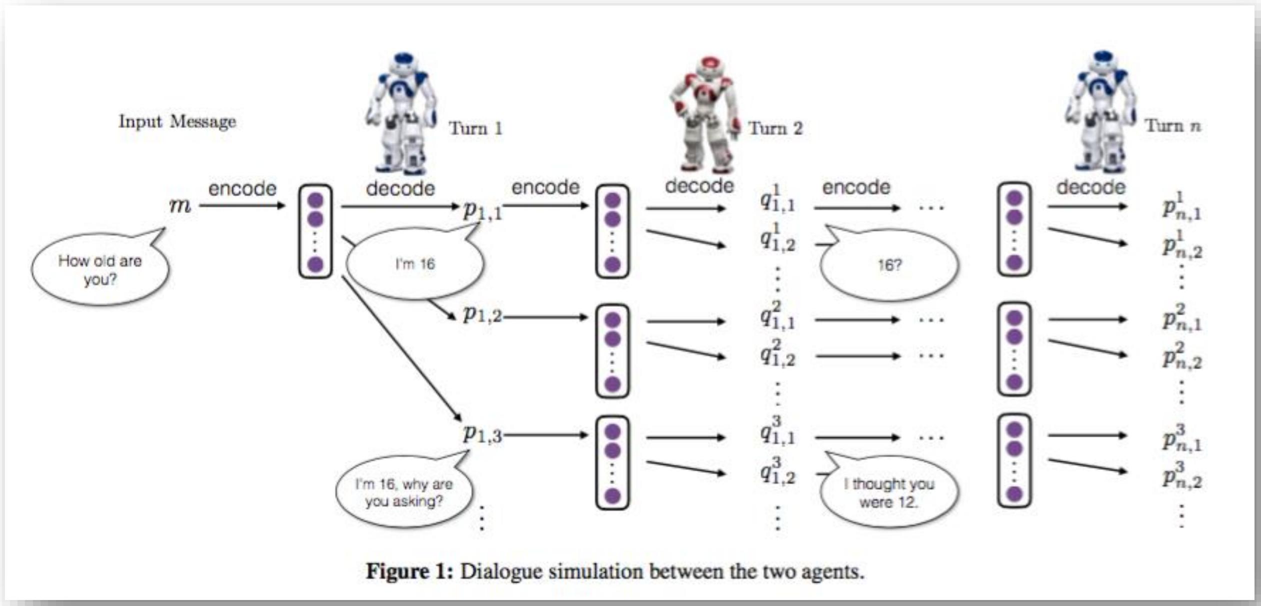


Figure 1: Dialogue Generation Over Time

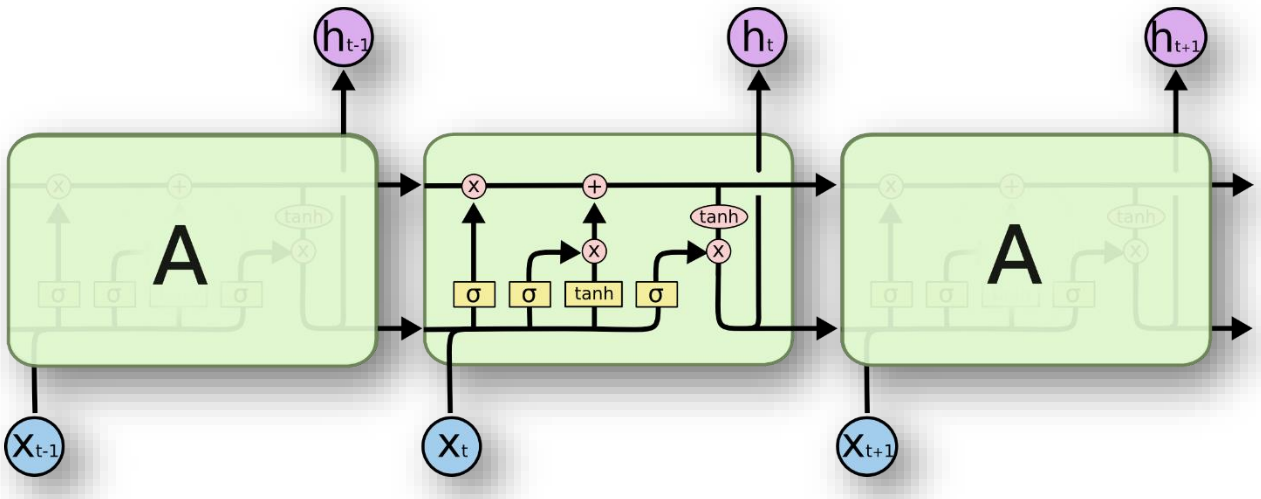


Figure 2: A Typical LSTM repeating module

1.4 Applications

Dialog schemes can support a wide range of applications in commercial, teaching, administration, healthcare, and entertainment. For instance:

- Replying queries about products and services through a business's website or intranet portal.
- **Client service agent information base:** Permits agents to type in a client's query and guide them with an answer
- **Guided advertising:** Enabling connections by providing responses and direction in the sales process, mainly for compound products being sold to greenhorn clients.
- **Service:** Replying to core operative queries, e.g., replying HR queries
- **Website triangulation:** Guiding clients to applicable percentages of compound websites—a Website custodian.
- **Technical support:** Answering to practical difficulties, such as diagnosing a problem with a invention or device.

2. LITRATURE SURVEY

2.1 Related Work

Statistical systems basically Fall into two major categories**.** First, treats the problem as a source-to-target transduction problem and learns mapping rules between input messages and responses from a massive amount of training data. The second one completely concentrates on building task-oriented dialogue systems to solve domain-specific tasks.

Ritter et al. (2011) frames the response generation problem as a statistical machine translation (SMT) problem. Sordoni et al. (2015) improved Ritter et al.'s system by rescoring the outputs of a phrasal SMT-based conversation system with a neural model that incorporates prior context. RecentprogressinSEQ2SEQ models inspire several efforts (Vinyals and Le, 2015) to build end-to-end conversational systems which first apply an encoder to map a message to a distributed vector representing its semantics and generate a response from the message vector. Serban et al. (2016) propose a hierarchical neural model that captures dependencies over an extended conversation history. Li et al. (2016a) propose mutual information between message and response as an alternative objective function to reduce the proportion of generic responses produced by SEQ2SEQ systems.

Efforts include statistical models such as Markov Decision Processes (MDPs), POMDP (Young et al., 2013; Gaˇsic et al., 2014) models, and models that statistically learn generation rules. This dialogue literature thus widely applies reinforcement learning to train dialogue policies. But then task-oriented RL dialogue classifications frequently depend on sensibly restricted dialogue limitations, or hand-built patterns with state, action and reward signals calculated by people for each new area, making the model difficult to spread to open-domain situations.

2.2 Reinforcement Learning for Open-Domain Dialogue

In this section, the components of the proposed RL model are described in detail. The learning system consists of two agents. **'p'** signifies sentences produced from the first agent and **'q'** signifies sentences from the second. The two agents take turns talking with each other. A dialogue can be represented as an alternating sequence of sentences generated by the two agents: **p1, q1, p2, q2..., pi, qi**. The generated sentences are viewed as actions that are taken according to a policy defined by an encoder-decoder recurrent neural network language model. The parameters of the network are optimized to maximize the expected future reward using policy search.

Policy gradient methods are more appropriate for this scenario than Q-learning, because the encoder-decoder RNN can be initialized using MLE parameters that already produce plausible responses, before changing the objective and tuning towards a policy that maximizes long-term reward. Q-learning, on the other hand, directly estimates the future expected reward of each action, which can differ from the MLE objective by orders of magnitude, thus making MLE parameters inappropriate for initialization. The components (states, actions, reward, etc.) of this sequential decision problem are summarized in the following sub-sections.

- **Action**

  An action **a** is the dialogue utterance to generate. The action space is infinite since arbitrary-length sequences can be generated.
- **State**

  A state is denoted by the previous two dialogue turns **[pi, qi]**. The dialogue history is further transformed to a vector representation by feeding the concatenation of **pi** and **qi** into an LSTM encoder model.
- **Policy**

  A policy takes the form of an LSTM encoder-decoder **(i.e., P_{RL} (P_{i+1}|P_i, Q_i))** and is defined by its parameters. A stochastic representation of the policy (a probability distribution over actions given states) is used. A deterministic policy would result in a discontinuous objective that is difficult to optimize using gradient-based methods.
- **Reward**

  'r' denotes the reward obtained for each action. In this subsection, the major factors that contribute to the success of a dialogue are discussed and approximations that describe these factors can be operationalized in computable reward functions.

2.3 Ease of answering

The aspect of a turn is related to its forward-looking function: the constraints a turn places on the next turn. The proposed technique measures the ease of answering a produced turn by means of the negative log likelihood of replying to that sound with a dull reply. A list of dull responses was manually constructed, **'S'** consisting 8 turns such as "I don't know what you are talking about", "I have no idea", etc., that has been found to occur very frequently in SEQ2SEQ models of conversations. The reward function is given as follows:

$$\text{r1} = -\frac{1}{N_S}\sum_{S \in s} \frac{1}{N_s} \log p_{Seq2Seq}(s|a)$$

where N_S denotes the cardinality of N_S and Ns denotes the number of tokens in the dull responses.

2.4 Information Flow

Each agent is needed to contribute new information at each turn to keep the dialogue moving and avoid repetitive sequences. Therefore, penalizing semantic similarity between consecutive turns from the same agent. Let h_{pi} and h_{pi+1} signify illustrations obtained from the encoder for two successive turns p_i and p_{i+1}. The reward is given by the negative log of the cosine similarity between them:

$$r_2 = -\log \cos(hpi, hp_{i+1})$$

2.5 Semantic Coherence

The adequacy of responses is also required to avoid situations in which the generated replies are highly rewarded but are ungrammatical or not coherent. Therefore, it is considered that the mutual information between the action **a** and previous turns in the history to ensure the generated responses are coherent and appropriate:

$$r_3 = \frac{1}{N_a} \, logP_{Seq2Seq}\,(a|q_i, p_i) + \frac{1}{N_{q_i}} logP_{Seq2Seq}^{backward}(q_i|a)$$

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

$$r(a, [p_i, q_i]) = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3$$

where $\lambda_1 + \lambda_2 + \lambda_3 = 1$ for $\lambda_1 = 0.25$, $\lambda_2 = 0.25$ and $\lambda_3 = 0.5$.

2.6 Some Sample Conversations:

Conversation 1: Basic

Human: hello!

Machine: hello!

Human: how are you?

Machine: I ’m good.

Human: what ’s your name?

Machine: I ’m Julia.

Human: when were you born?

Machine: July 20th.

Human: what year were you born?

Machine: 1977.

Human: where are you from?

Machine: I ’m out in the boonies.

Human: are you a man or a woman?

Machine: I ’m a woman.

Human: why are we here?

Machine: I ’m not sure.

Human: okay, bye!

Machine: bye.

Human: see you later.

Machine: bye.

Conversation 2: Simple Q&A (contexts and multiple choice)

Human: my name is David. what is my name?

Machine: David.

Human: my name is john. what is my name?

Machine: john.

Human: are you a leader or a follower?

Machine: I ’m a leader.

Human: are you a follower or a leader?

Machine: I ’m a leader.

Conversation 3: Philosophical Q&A

Human: what is the purpose of life?

Machine: to serve the greater good.

Human: what is the purpose of living?

Machine: to live forever.

Human: what is the purpose of existence?

Machine: to find out what happens when we get to the planet earth.

Human: where are you now?

Machine: I ’m in the middle of nowhere.

Human: what is the purpose of dying?

Machine: to have a life.

Human: what is the purpose of being intelligent?

Machine: to find out what it is.

Human: what is the purpose of emotions?

Machine: I don ’t know.

3. WORK AND CONTRIBUTIONS

3.1 Dataset Pre-processing

In this step dependencies between characters and the conditional probabilities of characters in sequences is learnt so that a wholly new and original sequences of characters can be generated. The dataset used is a book named **“Alice in Wonderland”**. Since the limitation in neural networks allow them to learn numbers only the strings and characters need to be encoded into integers with a special kind of encoding scheme which ought to be reversible to retrieve the original string back. This can be done by creating a set of all distinct characters in the book, then creating a map of each character to a unique integer. The book has just under 150,000 characters and when converted to lowercase there are only 47 distinct characters in the vocabulary for the network to learn.

Now the training data for the network was defined. The book text was split into subsequences with a fixed length of 100 characters. Each training pattern of the network comprises of 100-time steps of one character (X) followed by one-character output (y). When creating these sequences, this window was slide along the whole book one character at a time, allowing each character a chance to be learned from the 100 characters that preceded it. For example, if the sequence length is 5 (for simplicity) then the first two training patterns would be as follows:

1. CHAPT -> E
2. HAPTE -> R

After this step the list of input sequences were transformed into the form [samples, time steps, features] as expected by an LSTM network. The integers were rescaled to the range 0-to-1 to make the patterns easier to learn by the LSTM network. Finally, the output patterns (single characters converted to integers) were converted into a one hot encoding. This is to configure the network to predict the probability of each of the 47 different characters in the vocabulary (an easier representation) rather than trying to force it to predict precisely the next character. Each **y** value is converted into a sparse vector with a length of 47, full of zeros except with a 1 in the column for the letter (integer) that the pattern represents.

3.2 Developing a small LSTM neural Network

A single hidden LSTM layer is defined with **256 memory units**. The network uses dropout with a **probability of 20**. The output layer is a Dense layer using the **SoftMax** activation function to output a probability prediction for each of the **47 characters** between 0 and 1.

The problem is really a single character classification problem with **47 classes** and as such is defined as optimizing the **log loss (cross entropy)**, here using the **ADAM optimization algorithm** for speed. There is no test dataset. Hence the entire training dataset was modelled to learn the probability of each character in a sequence.

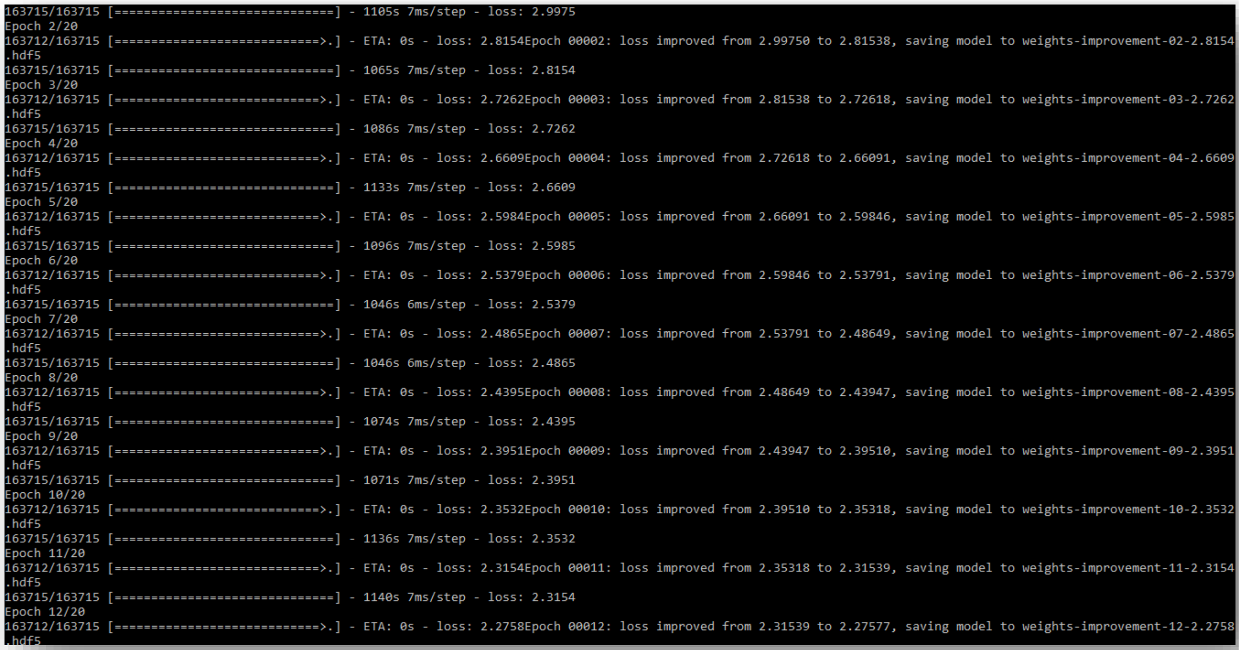
We are not interested in the most accurate (classification accuracy) model of the training dataset. This would be a model that predicts each character in the training dataset perfectly. Instead we are interested in a generalization of the dataset that minimizes the chosen loss function. A balance between generalization and overfitting is important but short of memorization.

The network is slow to train (about 275 milli-seconds per epoch on an Nvidia 940M GPU). Because of the slowness and because of the optimization requirements, we will model checkpointing is used to record all the network weights to file each time an improvement in loss is observed at the end of the epoch. The best set of weights (lowest loss) is saved to instantiate our generative model in the next section.

3.3 Generating text using an initial seed

Data was loaded, and the network was defined in a similar manner, except the network weights were loaded from a checkpoint file and the network did not need to be trained. Starting with a seed sequence as input, the next character was generated then the seed sequence was updated to add the generated character on the end and trim off the first character. This process was repeated for a sequence of 1,000 characters in length. A random input pattern was picked as the seed sequence, then the generated characters were printed as they were generated.

3.4 Results



```
163715/163715 [==============================] - 1105s 7ms/step - loss: 2.9975
Epoch 2/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.8154Epoch 00002: loss improved from 2.99750 to 2.81538, saving model to weights-improvement-02-2.8154
.hdf5
163715/163715 [==============================] - 1065s 7ms/step - loss: 2.8154
Epoch 3/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.7262Epoch 00003: loss improved from 2.81538 to 2.72618, saving model to weights-improvement-03-2.7262
.hdf5
163715/163715 [==============================] - 1086s 7ms/step - loss: 2.7262
Epoch 4/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.6609Epoch 00004: loss improved from 2.72618 to 2.66091, saving model to weights-improvement-04-2.6609
.hdf5
163715/163715 [==============================] - 1133s 7ms/step - loss: 2.6609
Epoch 5/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.5984Epoch 00005: loss improved from 2.66091 to 2.59846, saving model to weights-improvement-05-2.5985
.hdf5
163715/163715 [==============================] - 1096s 7ms/step - loss: 2.5985
Epoch 6/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.5379Epoch 00006: loss improved from 2.59846 to 2.53791, saving model to weights-improvement-06-2.5379
.hdf5
163715/163715 [==============================] - 1046s 6ms/step - loss: 2.5379
Epoch 7/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.4865Epoch 00007: loss improved from 2.53791 to 2.48649, saving model to weights-improvement-07-2.4865
.hdf5
163715/163715 [==============================] - 1046s 6ms/step - loss: 2.4865
Epoch 8/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.4395Epoch 00008: loss improved from 2.48649 to 2.43947, saving model to weights-improvement-08-2.4395
.hdf5
163715/163715 [==============================] - 1074s 7ms/step - loss: 2.4395
Epoch 9/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.3951Epoch 00009: loss improved from 2.43947 to 2.39510, saving model to weights-improvement-09-2.3951
.hdf5
163715/163715 [==============================] - 1071s 7ms/step - loss: 2.3951
Epoch 10/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.3532Epoch 00010: loss improved from 2.39510 to 2.35318, saving model to weights-improvement-10-2.3532
.hdf5
163715/163715 [==============================] - 1136s 7ms/step - loss: 2.3532
Epoch 11/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.3154Epoch 00011: loss improved from 2.35318 to 2.31539, saving model to weights-improvement-11-2.3154
.hdf5
163715/163715 [==============================] - 1140s 7ms/step - loss: 2.3154
Epoch 12/20
163712/163715 [============================>.] - ETA: 0s - loss: 2.2758Epoch 00012: loss improved from 2.31539 to 2.27577, saving model to weights-improvement-12-2.2758
.hdf5
```

Figure 1: Training the dataset for 20 epochs

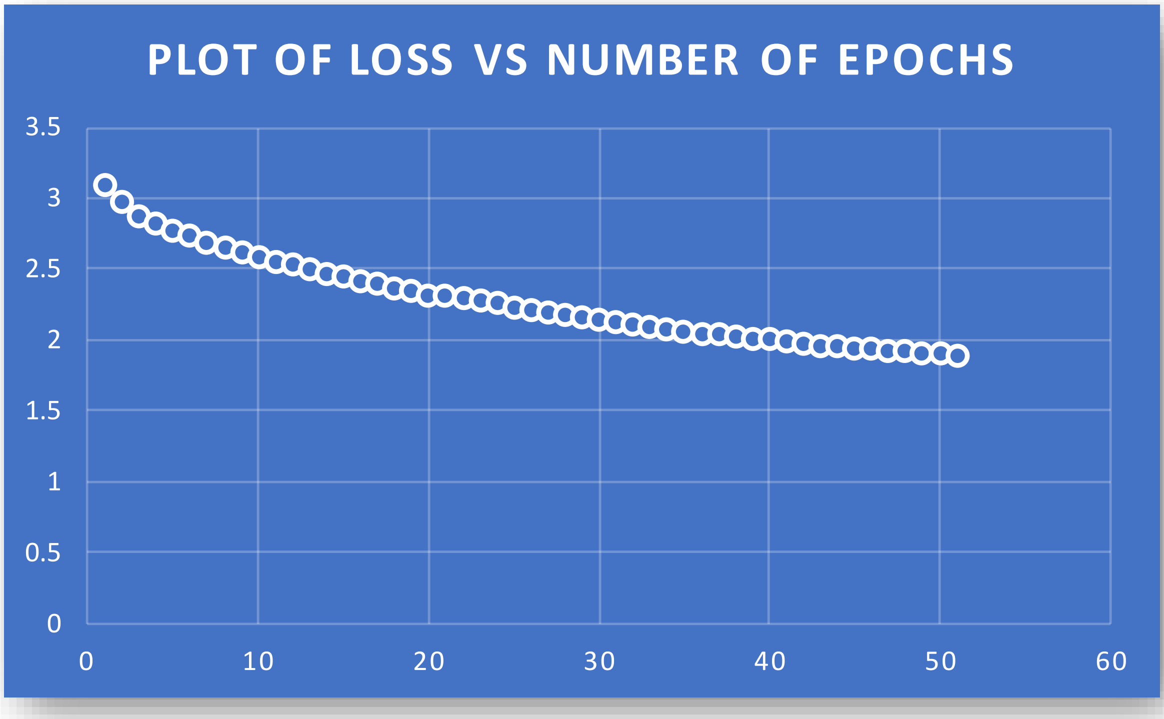
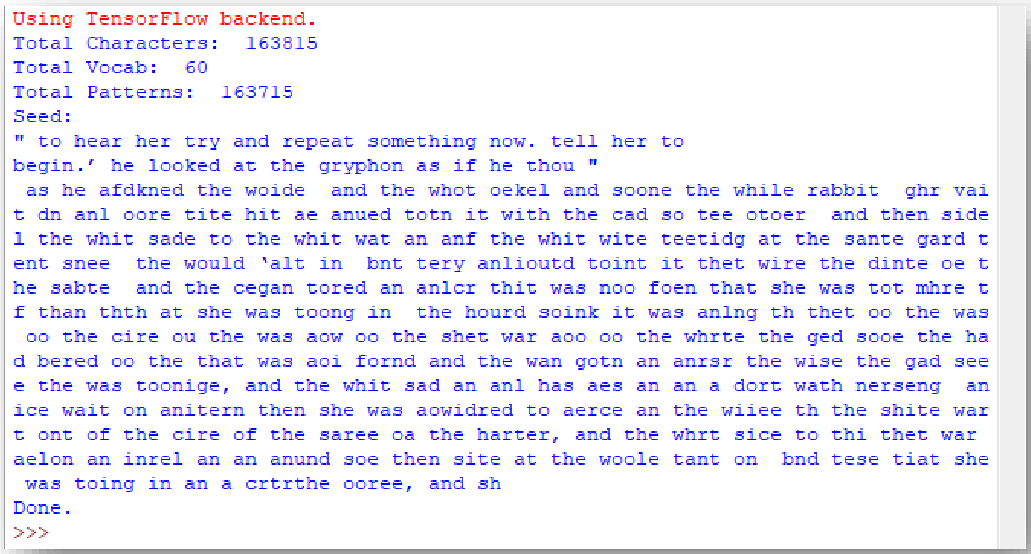


Figure 2: Plot of Loss vs Number of Epochs



```
Using TensorFlow backend.
Total Characters:  163815
Total Vocab:  60
Total Patterns:  163715
Seed:
" to hear her try and repeat something now. tell her to
begin.' he looked at the gryphon as if he thou "
 as he afdkned the woide  and the whot oekel and soone the while rabbit  ghr vai
t dn anl oore tite hit ae anued totn it with the cad so tee otoer  and then side
l the whit sade to the whit wat an anf the whit wite teetidg at the sante gard t
ent snee  the would 'alt in  bnt tery anlioutd toint it thet wire the dinte oe t
he sabte  and the cegan tored an anlcr thit was noo foen that she was tot mhre t
f than thth at she was toong in  the hourd soink it was anlng th thet oo the was
 oo the cire ou the was aow oo the shet war aoo oo the whrte the ged sooe the ha
d bered oo the that was aoi fornd and the wan gotn an anrsr the wise the gad see
e the was toonige, and the whit sad an anl has aes an an a dort wath nerseng  an
ice wait on anitern then she was aowidred to aerce an the wiiee th the shite war
t ont of the cire of the saree oa the harter, and the whrt sice to thi thet war
aelon an inrel an an anund soe then site at the woole tant on  bnd tese tiat she
 was toing in an a crtrthe ooree, and sh
Done.
>>>
```

Figure 3: Result after 50 Epochs with the initial seed

4. CONCLUSION

This report shows the generation of dialogues from an initial seed which can then be extended to a conversation that can be simulated to fit many applications. This is also a subproblem of NLP (Natural Language Processing) where semantics and coherence are the most important concern. But here we rather focus on the generation of the text with least log loss as discussed above. Further work can be added by implementing the simulation to two agent environments and then improve the semantics of the result provided above by further training.

REFERENCES

[1] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao and Dan Jurafsky, *"Deep Reinforcement Learning for Dialogue Generation"* 29th Sept 2016

[2] Ilya Sutskever, Oriol Vinyals, Quoc V. Le, *"Sequence to Sequence Learning with Neural Networks"* 14th Dec 2014

[3] Oriol Vinyals, Quoc V. Le, *"A Neural Conversational Model"* 22nd July 2015

[4] Alex Graves, *"Generating Sequences with Recurrent Neural Networks"* 5th June 2014