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 necessity that directed traditional NLP models of dialogue to use 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 also human juries screening that the algorithm produces more interactive replies than the previous generation Neural Network Models.

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 among two simulated agents to discover the state of conceivable 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 likelihood of producing a response given the prior 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 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 repetitive replies.

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 likelihood of producing a response given the prior 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 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 repetitive replies.

1.3 Illustrations

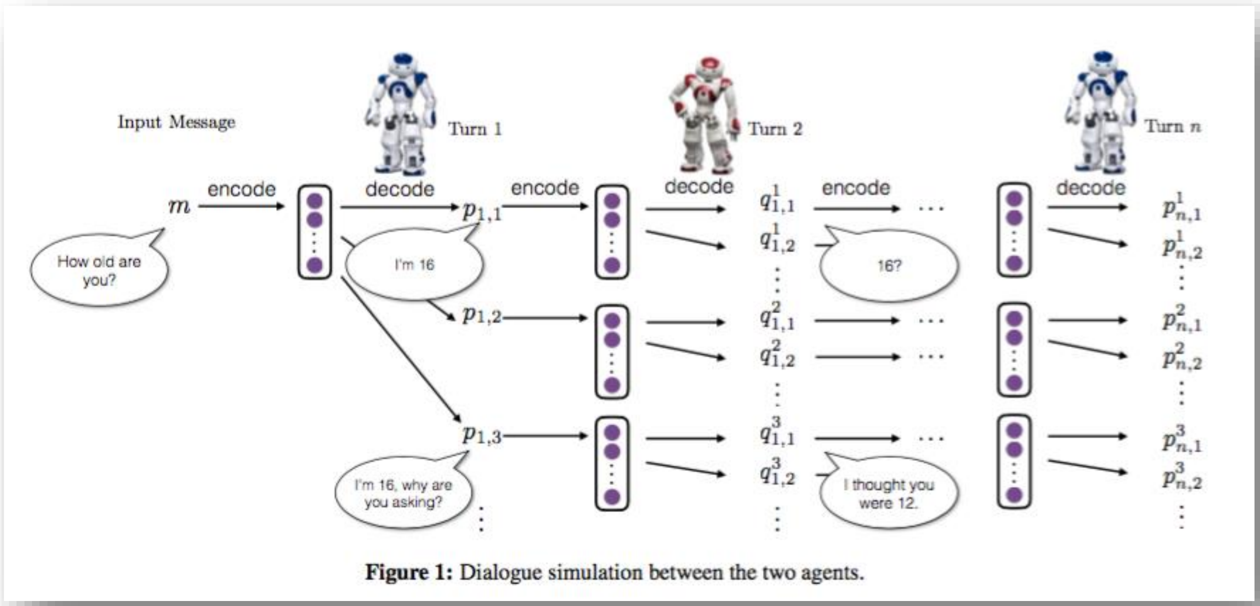


Figure 1: Dialogue Generation Over Time

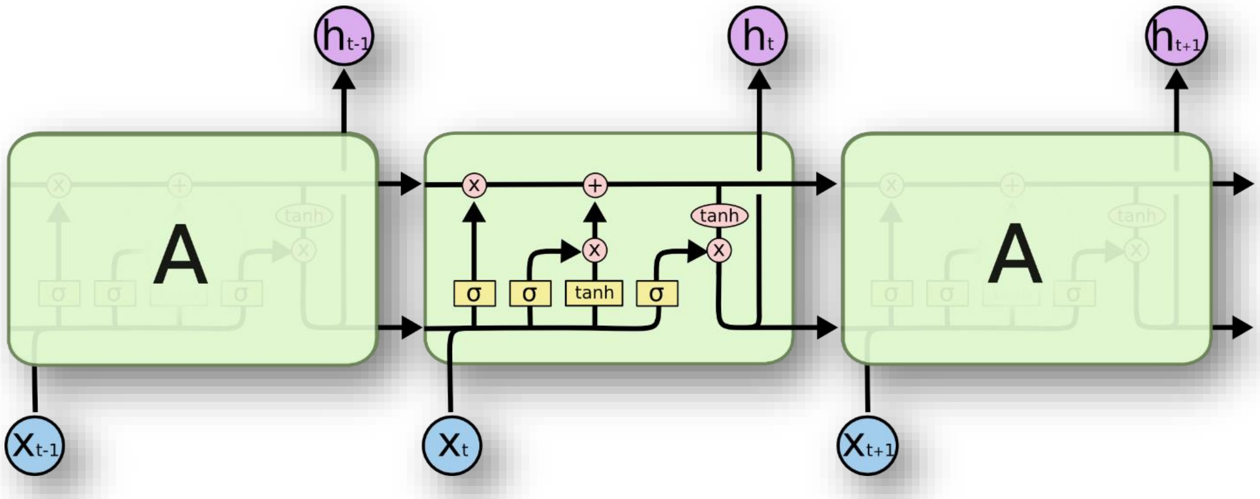


Figure 2: A Typical LSTM repeating module

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 Drop into two main groups**.** First, treats the problem as a source-to-target transduction problem and absorbs mapping rules among input messages and replies from a huge quantity of training data. The second one completely concentrates on making task-oriented dialogue systems to resolve domain-specific tasks.

Ritter et al. (2011) frames the response generation problem as a statistical machine translation (SMT) problem. Sordoni et al. (2015) enhanced Ritter et al.'s scheme by rescoring the yields of a linguistic SMT-based dialog system with a neural model that incorporates prior context. Current development in SEQ2SEQ models motivate numerous efforts (Vinyals and Le, 2015) to build end-to-end conversational systems that apply an encoder to encode a message to a distributed vector representing its semantics and produce a reply from the message vector. Serban et al. (2016) suggest a hierarchical neural model that captures dependencies over a long chat history. Li et al. (2016a) suggest shared data between message and reply as a substitute objective function to decrease the amount of generic replies formed by SEQ2SEQ systems.

Efforts contain arithmetic models like Markov Decision Processes (MDPs), POMDP (Young et al., 2013; Gaˇsic et al., 2014) models, and models that statistically pick up cohort guidelines. This dialogue work thus extensively applies reinforcement learning to train dialogue rules. 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 Dialogue Generation

The components of the proposed RL model are described in detail. The learning scheme contains two agents. **'p'** signifies verdicts produced from the first agent and **'q'** from the second. A dialogue can be characterized as an irregular order 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 rule defined by an encoder-decoder recurrent neural network language model. The constraints of the network are improved to maximize the probable future reward using rule search.

Policy gradient methods are more suitable for this situation than Q-learning, because the encoder-decoder RNN can be initialized using MLE parameters that previously produce reasonable replies, before altering the objective and tuning to a rule that exploits long-term reward. Q-learning, directly approximations the future predictable reward of each action, that may fluctuate from the MLE objective by orders of scale, hence making MLE parameters unsuitable for initialization. The components of this successive decision problem are:

- **Action**

  An action **a** is the dialogue utterance to produce. The action space is endless as random-length sequences can be produced.
- **State**

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

  A policy is 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 is used.
- **Reward**

  'r' signifies the reward found for each action. The major issues that contribute to the success of a dialogue are shown.

2.3 Ease of answering

The feature of a chance is linked to its forward-looking function: the limits a chance places on the next chance. The proposed method calculates the ease of answering a produced chance by the help of the negative log likelihood of responding to that sound with a dull answer. A list of dull answers was manually constructed, **'S'** consisting 9 chances such as "I don't know", "I have no idea what are you talking about??", etc., that has been found to arise very often in SEQ2SEQ models of conversations. The reward function can be represented:

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

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

2.4 Information Flow

Each agent is needed to contribute new info at each chance to maintain the dialogue flow and avoid repetitive arrangements. Hence, penalizing semantic resemblance between successive choices from the same agent. Let h_{pi} and h_{pi+1} signify illustrations obtained from the encoder for two consecutive choices p_i and p_{i+1}. The reward is the negative log of the cosine comparison between them:

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

2.5 Semantic Coherence

The suitability of replies is also required to evade conditions in which the produced replies are extremely rewarded but are faulty or not clear. Therefore, it is considered that the common info among the action **a** and previous choices in the past to confirm the produced replies are clear and suitable:

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

The final reward for action a is the 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 $\lambda_3 = 0.5$.

3. WORK AND CONTRIBUTIONS

3.1 Dataset Pre-processing

In this step dependencies among characters and the conditional probabilities of characters in sequences is learnt as 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 list of all different characters in the book, then constructing a map of each character to a unique integer. The book has 163,815 characters and when transformed to lowercase there are 46 unique characters in the terminology for the network.

Now the training data for the network was defined. The book was divided into subsequence’s of fixed length of 95 characters. The training pattern of the network consists of 98-time steps of each character (X) followed by one-character output (y). While generating these arrangements, this window was slide along the whole book one character at a time, permitting each character a chance to be learned from the 100 characters that go before it. If the sequence length is 4 then two training patterns would be as follows:

1. HITL -> E
2. ITLE -> R

After this step the set of input sequences were transformed to [samples, time steps, features] as expected by an LSTM network. The integers were rescaled to the range 0-to-1 to make the patterns simpler to learn by the LSTM network. The output patterns were converted into one hot encoding format. This is to configure the network to predict the likelihood of each of the 46 dissimilar characters in the language rather than forcing it to predict next character precicely.

3.2 Developing a small LSTM neural Network

A single LSTM layer is defined with **256 memory units**. The network uses dropout with a **likelihood of 20**. The output layer uses the **SoftMax** activation function to generate a likelihood prediction for each of the **46 characters** between 0 and 1.

The problem is a single character classification with **46 classes** and is defined as improving the **log loss (cross entropy)** utilizing the **ADAM optimization algorithm** for speed. No Test-Dataset is available, so the complete training dataset was modelled to learn the likelihood of each character in a sequence.

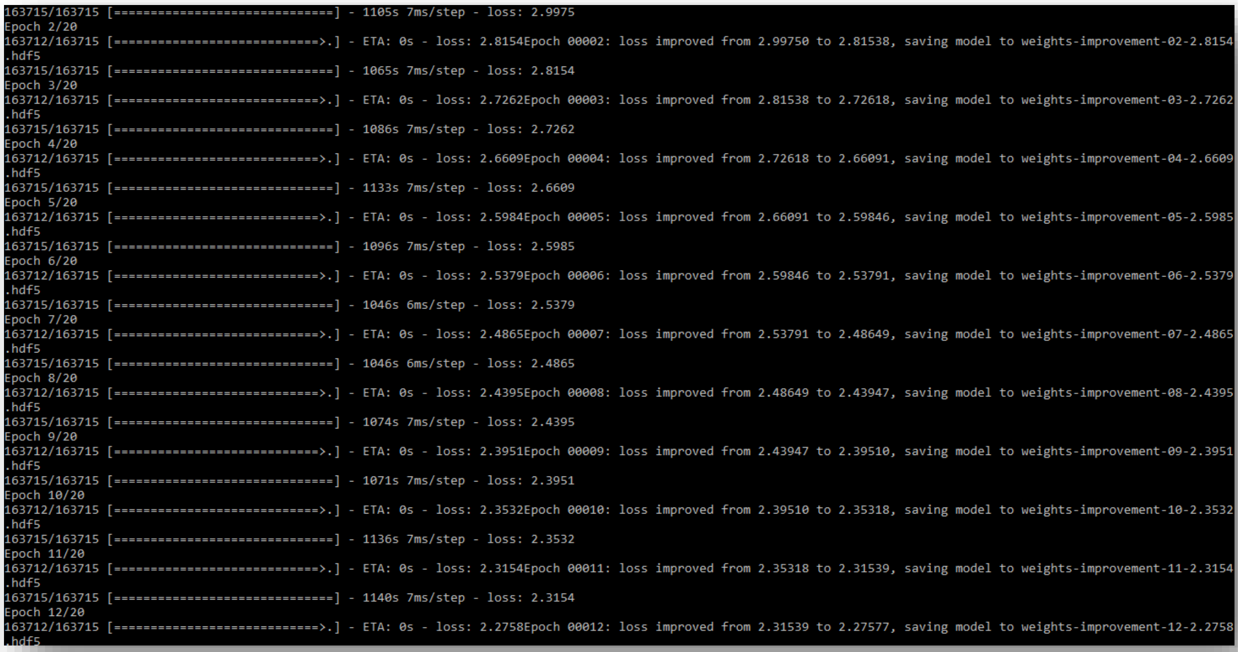
Here we are concerned in a generalization of the dataset that reduces the chosen loss function. An equilibrium between generalization and overfitting is significant.

The network is slow to train (about 275 milli-seconds per epoch on an Nvidia 940M GPU). Because of the slowness and optimization necessities, 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, but 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

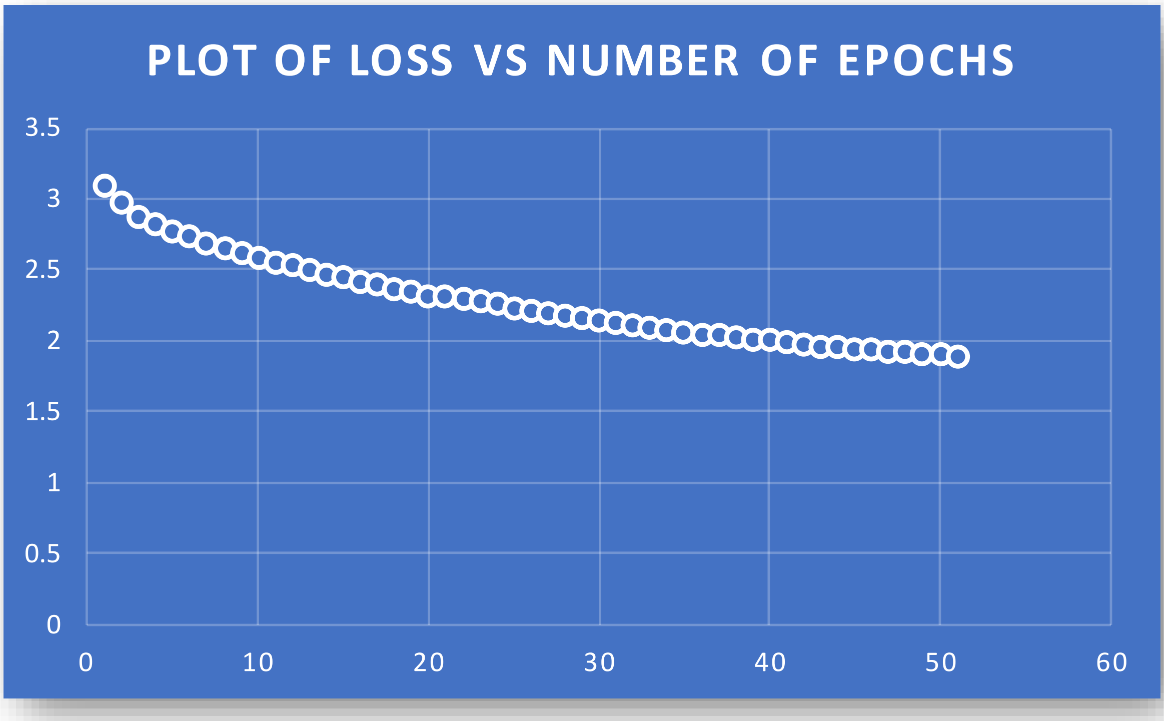
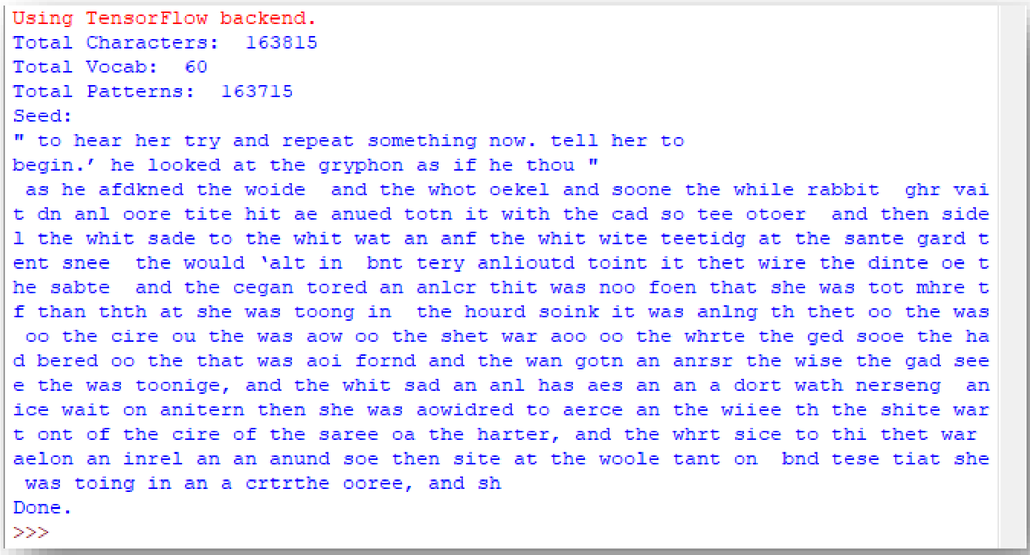


Figure 2: Plot of Loss vs Number of Epochs



```
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