Story Generator

by

Talatala Rahul Reddy 19BCE1778

Pulimi Bhargava Reddy 19BCE1342

A project report submitted to

Dr. Bharadwaja Kumar

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

in partial fulfillment of the requirements for the course of

CSE4022 – Natural Language Processing

in

B. Tech. COMPUTER SCIENCE ENGINEERING

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

Vandalur – Kelambakkam Road

Chennai – 600127

December 2021

BONAFIDE CERTIFICATE

Certified that this project report entitled "**Project Name"** is a bonafide work of **TALATALA RAHUL REDDY – 19BCE1778 and PULIMI BHARGAVA REDDY – 19BCE1342** who carried out the project work under my supervision and guidance for **CSE4022 – Natural Language Processing.**

Dr. Bharadwaja Kumar

Associate Professor

School Of Computer Science & Engineering (SCOPE),

VIT University, Chennai

Chennai – 600 127.

ABSTRACT

Machines have become so powerful and quick that they can weave short stories into technically rich blogs and huge novels. Machines are capable of producing words with contextual understanding more than ever. This is just conceivable with the assistance of NLP. Simply because of the progression of NLP machines can comprehend the specific situation and twist up stories without help from anyone else. It uses information from computational linguistics and artificial intelligence to create natural language text that can meet specific communication needs.

We would like to build a model with the assistance of NLP tools & libraries which would help us to create rich stories automatically. Text generation is a branch of natural language processing. It has a wide variety of applications in automatic documentation systems like automatic letter writing, automatic report generation, etc.

ACKNOWLEDGEMENT

We wish to express our sincere thanks and deep sense of gratitude to our project guide, **Dr. Bharadwaja Kumar,** Associate Professor, School of Computer Science & Engineering, for his consistent encouragement and valuable guidance offered to us in a pleasant manner throughout the course of the project work.

We are extremely grateful to **Dr. Jagadeesh Kannan R,** Dean of School of Computer Science & Engineering, VIT Chennai, for extending the facilities of the School towards our project and for his unstinting support.

We also take this opportunity to thank all the faculty of the School for their support and their wisdom imparted to us throughout the course.

TABLE OF CONTENTS

Keywords:

Text generation, recurrent neural networks, GRU, Adversarial training Machine translation, Deep learning, Natural language generation, LSTM networks, Sequence models, language models

Introduction:

Natural language processing (NLP) has advanced rapidly in recent years, owing to advances in learning-based algorithms and increases in CPU capacity. As computational resources became more readily available, active research and development in the architecture of deep learning algorithms occurred. Although feed-forward neural networks have produced amazing results in a variety of applications, they are not regarded best for language modelling. This is due to the fact that the feed-forward network strives to maximise the input-output function regardless of the order of the inputs. The sequence of encoding is crucial in Natural language processing because the type of data employed (both speech and text) is akin to time-series data.

The ability to generate text that is near to the quality of human-generated writing has a wide range of applications, including language translation, chatbots, and question answering, among others. Maximum likelihood estimate has proven that RNN extensions such as LSTMs or GRUs that retain long-term memory of tokens function well in practise...

a little introduction to RNN:

A recurrent neural network (RNN) is a type of sequence-based neural network with memory units that can recall past data. Only the output of the previous layers is sent into the following layer in feed-forward networks. The output of each cell (neuron) in RNNs is delivered back to the same cell as input, thus the name "recurrent networks." The positioning information of words in sequences is recorded as delays in the memory unit due to its recurring nature. This assists the recurrent network in maintaining word order. The current state of the RNN cell is determined using the current cell's input and the previous state's output. As a result, in addition to the preceding layer's output, the current layer's output at the prior timestamp is provided as input to the current layer.

(a) RNN in state representation

(b) unfolded RNN sequence

The architecture of a vanilla RNN in state representation and the representation of the same RNN when unfolded based on time information are shown in the diagram above. The vanilla network is a term used to describe a generic model that has no specific cases or modifications. The "xt" block denotes the input, the "ht" block denotes the output, and the "A" block denotes the hidden layer with a single activation function.

LSTM:

Short-term memory is a problem for recurrent neural networks. They'll have a hard difficulty transporting information from earlier time steps to later ones if the sequence is long enough. If you're trying to predict something from a paragraph of text, RNNs may leave out critical information at the start.

The vanishing gradient problem affects recurrent neural networks during back propagation. Gradients are values that are used to update the weights of a neural network. When a gradient reduces as it back propagates through time, this is known as the vanishing gradient problem. When a gradient value falls below a certain threshold, it no longer contributes much to learning.

Layers that get a modest gradient update in recurrent neural networks stop learning. Those are frequently the first layers to appear. RNNs can forget what they've seen in longer sequences because these layers don't learn, resulting in a short-term memory.

As a solution to short-term memory, LSTMs and GRUs were developed. They have inbuilt devices known as gates that can control the flow of data.

These gates can figure out which data in a sequence should be kept and which should be discarded. It can then transfer important information down the long chain of sequences to make predictions as a result of this. These two networks are responsible for nearly all state-of-the-art recurrent neural network findings. Voice recognition, speech synthesis, and text production all use LSTMs and GRUs.

hidden state computation, To begin, a vector is created by combining the input and prior concealed state. The current and prior inputs are now represented in that vector. The vector is passed through the tanh activation, and the result is the network's new hidden state or memory.

Tanh activation is used to control the values that flow across the network. The tanh function compresses values so that they are always between -1 and 1.

Tanh squishes values in the range of -1 to 1.

When vectors pass through a neural network, they go through a number of changes as a result of various math operations.

The cell state and its many gates are at the heart of LSTMs. The cell state serves as a transportation highway for relative information as it travels down the sequence chain. You might think of it as the network's "memory."

As a result, information from earlier time steps can make its way to later time steps, lessening the short-term memory effects. Information is added or withdrawn from the cell state via gates as the cell state travels. The gates are several neural networks that determine which information about the cell state is permitted. During training, the gates might learn what information is important to keep or forget. In theory, the cell state can carry relevant information throughout the sequence's processing.

Sigmoid activations are found in gates. Tanh activation is comparable to sigmoid activation. Instead of squishing numbers ranging from -1 to 1, it squishes values ranging from 0 to 1. Because every integer multiplied by 0 equals 0, values vanish or are "lost," this is useful for updating or forgetting data. Because any number multiplied by one has the same value, the value remains the same or is "preserved." The network can figure out which data is unimportant and should be deleted, and which data should be kept. Let's forget about input, cell, and output gates...

Forget gate:

The first is the forget gate. This gate determines whether information should be discarded or saved. The sigmoid function passes information from the previous hidden state as well as information from the current input. The results are between 0 and 1. The closer you go to 0, the more you forget, and the closer you get to 1, the more you keep.

Input Gate

The input gate is used to update the cell state. First, we use a sigmoid function to combine the previous concealed state and the current input. By changing the values to be between 0 and 1, this determines which values will be updated. A value of 0 indicates that it is not important, whereas a value of 1 indicates that it is important. To help regulate the network, you also send the hidden state and current input into the tanh function to squish values between -1 and 1. The sigmoid result is then multiplied by the tanh output. The sigmoid output will determine which information from the tanh output should be kept.

Cell State

We should now have enough information to calculate the status of the cell. The cell state is first multiplied by the forget vector in a pointwise manner. If multiplied by values close to 0, this has the potential to drop values in the cell state. Then we conduct a pointwise addition on the output of the input gate, which updates the cell state to new values that

the neural network finds important. As a result, we now have a new cell state.

Output Gate

Last but not least, there's the output gate. The hidden state's next hidden state is determined by the output gate. It's important to remember that the concealed state contains data from earlier inputs. Predictions are also made using the hidden state. First, we use a sigmoid function to combine the previous concealed state and the current input. The newly adjusted cell state is then passed to the tanh function. To determine what information the hidden state should contain, we multiply the tanh output with the sigmoid output. The concealed state is the output. After that, the new cell state and concealed are carried over to the next time step.

The GRUs abandoned the cell state in favour of using the hidden state to transfer data. There are only two gates on it: a reset gate and an update gate.

Update Gate

The update gate functions similarly to an LSTM's forget and input gates. It determines what information should be discarded and what should be included.

Reset Gate:

The reset gate is another gate that is used to decide how much past information to forget.

And that’s a GRU. GRU’s has fewer tensor operations; therefore, they are a little speedier to train than LSTM’s.

Dataset:

a website brimful of stories. Link is provided below,

stories

Methodology:

The above dataset will be used for this task. Punctuation is removed during data preprocessing and only sentences of 7 or more words are used for training. The total number of training sets is also limited to 100,000 to allow control of the compute memory space.

I / O training instances are generated from the dataset. All words are vectorized using one-hot coding so that each word is represented by a vector of dimension V. Where V is the size of the vocabulary. The length of the input sequence, excluding the context vector, is set to 8. The training pattern context vector is also a V-dimensional vector. The process of coding the context vector depends on the extraction method. As mentioned, this is an all-to-one configuration, where the input contains a context vector to which the input word is added, and the output contains the next word in the sequence. The sequential model of the LSTM network applies in a configuration of bidirectional LSTMs, 256 hidden units, an inner layer "relu" activation, and an outer layer "softmax" activation. The Keras library is used to train the model. Since the language model is a predictive problem, that is, the model needs to predict one-hot coded target words, "softmax" is suitable for determining the maximum probability of a word. Implementation is done in Python using libraries such as Scikit-learn, NumPy, and Gensim. When the model is trained with the context vector, the prediction phase requires two sets of input from the user. The first input is a sequence of seed words. The seed word is the start word given to the network to start the text creation process. Since we trained the network with an input sequence of length 8, the network expects eight start words. The second set of inputs is the context word. You can provide

any number of context words as needed. The network produces a C sentence for a C context word. In the prediction phase, the network uses the first context word along with the first starting word to assemble the input and predict subsequent words. When a word is predicted, the first word is removed from the starting word, the predicted word is added, and the sequence length is maintained at 8. The next word is predicted using the same context word as the new input sequence. This process continues until the penalty is over. Completion of the sentence is indicated by dots. Once a sentence is completed, the next context word is used for generating the second sentence following the same procedure but instead of seed words, the words generated from the previous sentence are considered.

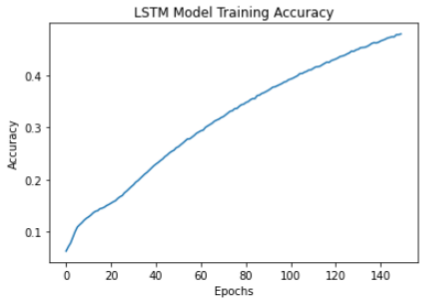
Output:

LSTM:

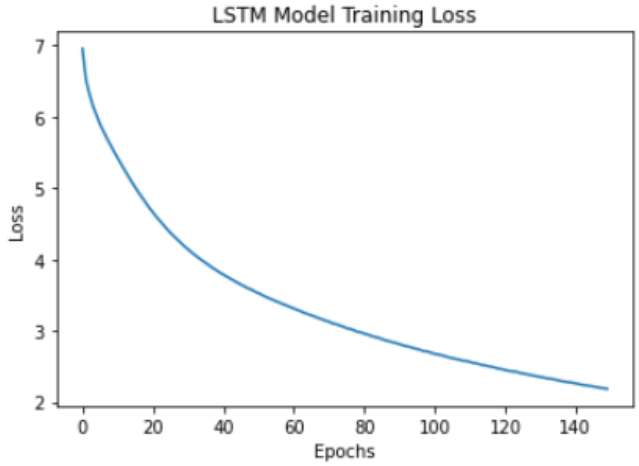
```
430/430 [==============================] - 23s 54ms/step - loss: 2.1938 - accuracy: 0.4775
Epoch 150/150
430/430 [==============================] - 23s 54ms/step - loss: 2.1851 - accuracy: 0.4789
```

```
In [ ]: 1 lstm.save('./LSTM.h5')
```

```
In [ ]: 1 import matplotlib.pyplot as plt
        2 epochs = range(len(lstm_history.history['accuracy']))
        3 plt.plot(epochs,lstm_history.history['accuracy'])
        4 plt.title("LSTM Model Training Accuracy")
        5 plt.xlabel("Epochs")
        6 plt.ylabel('Accuracy')
        7 plt.show()
```



In [ ]:
```
plt.plot(epochs,lstm_history.history['loss'])
plt.title("LSTM Model Training Loss")
plt.xlabel("Epochs")
plt.ylabel('Loss')
plt.show()
```



In [ ]:
```
seed_text = "The country was in chaos but"
generate_story(lstm, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[39]:
```
'something and corrupt one of the gallery she was clerk but that could spare that the baroness von koëldwethout of grogzwig he had bought the spirit of the eastern stars was a monotonous heartis to perspective vehemence and tables were for the change the park kicking the anchors hit the'
```

In [ ]:
```
seed_text = "I walked out of the store dissatisfied and it"
generate_story(lstm, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[40]:
```
'did he said splendid morning money fortifying course the health of the winning ticket that they ought me to play laying the baron s housekeeping and realized that he was gazing at the smooth greyness of the ride in all sorts of devotion was the guests of a radish curtains'
```

In [ ]:
```
seed_text = "The lady was known as a soothsayer - a psychic of sorts. The rubbing of her crystal ball was finalized as she c
generate_story(lstm, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[41]:
```
'to her hair in the tepid hospitality she do that it s a degree that i ve got in the march of manifest willing down tomorrow he looked at the viewless seem to you and be a little commonsense that is beyond the elms and i fairly a day got'
```

In [ ]:
```
seed_text = "Little Chandler's thoughts ever since lunch-time had been of his meeting with Gallaher, of Gallaher's invitatio
generate_story(lstm, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

Out[42]:
```
'he was ashamed of the awful however answered him to say to me all that he was bald and mabel as addressed me and close to me and tightening his knee he had not followed it and moment he said aloud he bowed and the guests commander are i ll see you a consulgeneral i was not going to hammer her cough i am sorry for irritation to r w d be any condition for you begged to re the author of the bible to the bath and morning again began to remain into the magazinesi next month is nt'
```

In [ ]:
```
seed_text = "On the morning of the fifth of September, in uniform, his revolver on the table, the colonel addressed his soli
generate_story(lstm, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

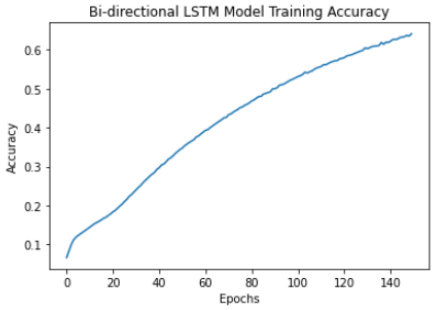
Out[43]:
```
'beside it and does nt matter that it was impossible for me to say why i ll get you a nightingale s nest that i ve got you informed you i ll have not my sick hunting i have got to my castle of danger i ve got meat thanks to me and demanded again on the doorstep in the surf silk and partly lyin up and on the high cigarette that you can not allow me to leave the table she said what i must be back for you and i said to go abroad for the baron who'
```

Bi directional LSTM:

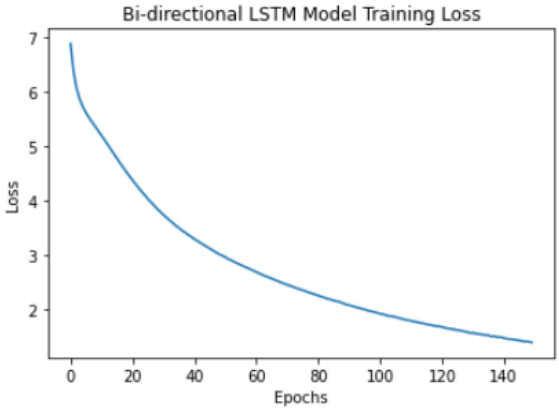
```
Epoch 150/150
430/430 [==============================] - 31s 72ms/step - loss: 1.3834 - accuracy: 0.6409
```

In [ ]:
```
bi_di_lstm.save('./Bi_di_LSTM.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt
epochs = range(len(bidi_lstm_history.history['accuracy']))
plt.plot(epochs,bidi_lstm_history.history['accuracy'])
plt.title("Bi-directional LSTM Model Training Accuracy")
plt.xlabel("Epochs")
plt.ylabel('Accuracy')
plt.show()
```



In [ ]:
```
plt.plot(epochs,bidi_lstm_history.history['loss'])
plt.title("Bi-directional LSTM Model Training Loss")
plt.xlabel("Epochs")
plt.ylabel('Loss')
plt.show()
```



Generating Stories from Bi-directional LSTM Model

In [ ]:
```
seed_text = "The country was in chaos but"
generate_story(bi_di_lstm, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[32]:
```
'the basin and the breathing observed in lace and pearls was struggling to flagstone to flagstone every sidewalk them and as if
he dared not go to words the principal went to her head and the lights he could see that the polar regions in a june thaw it wa
s'
```

In [ ]:
```
seed_text = "I walked out of the store dissatisfied and it"
generate_story(bi_di_lstm, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[33]:
```
'looks into the stump from a chameleonlike aptitude at impressions and swimming and his room the crystal the silver the chinath
ey were eating borsch the rich red soup with whipped cream so dear to russian palates half apologetically general zaroff i am a
fraid senator you had a good slim gentleman'
```

In [ ]:
```
seed_text = "The lady was known as a soothsayer - a psychic of sorts. The rubbing of her crystal ball was finalized as she c
generate_story(bi_di_lstm, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[34]:
```
'paul sank into the ooze he tried to manhood with children that he had a strange roseate faint green delicacy in the hedge she
were of snuff he was either a cupboard of wild file and ease with their arrival about him in the soapy water and he distinctly
heard'
```

In [ ]:
```
seed_text = "Little Chandler's thoughts ever since lunch-time had been of his meeting with Gallaher, of Gallaher's invitatio
generate_story(bi_di_lstm, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

Out[35]:
```
'questioned the crackling of the sea among fancy the blond fur of his overcoat his drivers of the mother better counsels prevai
led and a fragment of variegated quartz a row of mud shacks at the end of the wharf a straight dollars under the lodge and the
yellow moments of the rounding sun some yacht the room touched and every time he wore a towel and brandy and food of my bedroom
and they could watch him approach and keeping it back of the supper and the greater walk of the pantry and deserted she had wak
ed from a cold'
```

In [ ]:
```
seed_text = "On the morning of the fifth of September, in uniform, his revolver on the table, the colonel addressed his soli
generate_story(bi_di_lstm, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

Out[36]:
```
'a matter which the passing of a wild gallop and a row of mud shacks into the direction of the toothless old woman when the gia
nt his eyes descended they were smiling over the candlebox half from his brother s brain it prevented him half holding him to h
is white shadow when it was uncomfortably at the back of the situation or trees the wind shot their noses and looked into the g
ate and ran into the direction of the exit gate the hired car was still unfortunately before the dripping spiggots her human do
ubt going along and down at'
```

GRU:

```
430/430 [==============================] - 22s 51ms/step - loss: 0.7517 - accuracy: 0.7843
Epoch 150/150
430/430 [==============================] - 22s 51ms/step - loss: 0.7493 - accuracy: 0.7850
```

In [ ]:
```
gru.save('./GRU.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt
epochs = range(len(gru_history.history['accuracy']))
plt.plot(epochs,gru_history.history['accuracy'])
plt.title("GRU Model Training Accuracy")
plt.xlabel("Epochs")
plt.ylabel('Accuracy')
plt.show()
```

In [ ]:
```
plt.plot(epochs,gru_history.history['loss'])
plt.title("GRU Model Training Loss")
plt.xlabel("Epochs")
plt.ylabel('Loss')
plt.show()
```

Generating Stories from GRU Model

In [ ]:
```
seed_text = "The country was in chaos but"
generate_story(gru, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[32]:
```
'downcast he got down to her eyes and think that do they get to eating in rubbing fish while mother that bound when they bring
him round the doorstep with coatcollar turned out and then then it went out on the bed and went out rather arms and saw the'
```

In [ ]:
```
seed_text = "I walked out of the store dissatisfied and it"
generate_story(gru, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[33]:
```
'had suffered away in any mood s little times violent son and drove out a quarter of a deep black holes off the swamp beyond th
e deep fields of gate the phalanx of women in their front only fire in which they had passed the difficulty of interest and dri
nking'
```

In [ ]:
```
seed_text = "The lady was known as a soothsayer - a psychic of sorts. The rubbing of her crystal ball was finalized as she c
generate_story(gru, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[34]:
```
'the panther looked into one of silence when the doctor said she had heard his folly and liked his mind without discomfort was
a few weeks there and the doctor later crying and switching up his every morning and sat out her dick had sat between his aston
ished and stumpy'
```

In [ ]:
```
seed_text = "Little Chandler's thoughts ever since lunch-time had been of his meeting with Gallaher, of Gallaher's invitatio
generate_story(gru, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

Out[35]:
```
that was unnecessary to have at all their own people of the graceful enchanting man buildings and as she had made the carcass
at the stranger asked as he observed them that it would be effort to accept the own to force to pieces a wife with hot to dear
seen to tell what his dear fellow of a sledgehammer that she had told me now because it s rather to queen at this years he had
only used to give him a kitten mamma is at any service of your own daughter well just hearing of its service s'
```

In [ ]:
```
seed_text = "On the morning of the fifth of September, in uniform, his revolver on the table, the colonel addressed his soli
generate_story(gru, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

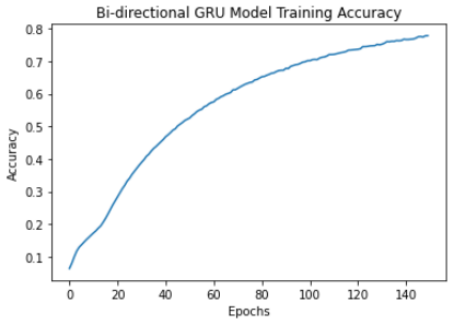
Out[36]:
```
'friends with pearls but making its discouraged gray dogs his suppers the mouth of his mouth as he remembered that he must lay
no voice gasping for look about from this he was climbed behind the people out of you do nt be alarmed said the sea gnashing it
s hands they were standing in the right to each other six minutes before he was sitting there in the air she was dressed in the
lightness of a short corridor glass there on the handle of the table there was no longer the fact and warm that there were no t
ime'
```

Bi directional GRU:

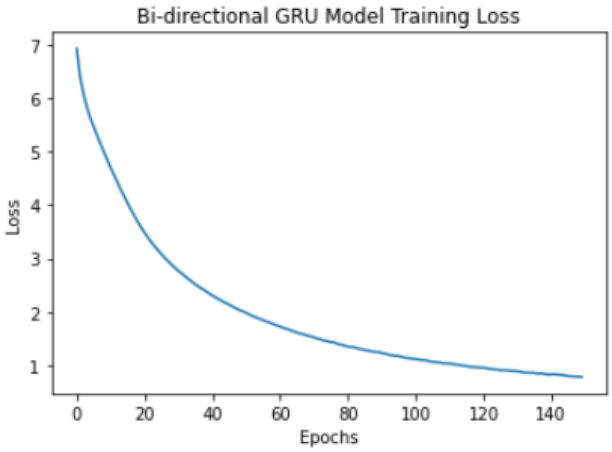
```
Epoch 150/150
430/430 [==============================] - 29s 68ms/step - loss: 0.7849 - accuracy: 0.7783
```

In [ ]:
```
bi_di_gru.save('./Bi_di_GRU.h5')
```

In [ ]:
```
import matplotlib.pyplot as plt
epochs = range(len(bidi_gru_history.history['accuracy']))
plt.plot(epochs,bidi_gru_history.history['accuracy'])
plt.title("Bi-directional GRU Model Training Accuracy")
plt.xlabel("Epochs")
plt.ylabel('Accuracy')
plt.show()
```



In [ ]:
```
plt.plot(epochs,bidi_gru_history.history['loss'])
plt.title("Bi-directional GRU Model Training Loss")
plt.xlabel("Epochs")
plt.ylabel('Loss')
plt.show()
```



Generating Stories from Bi-directional GRU Model

In [ ]:
```
seed_text = "The country was in chaos but"
generate_story(bi_di_gru, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[32]: 'they had helped they were back for a mask when he remembered was the child to it had a small patch of congealed blood and the dead s company was letting him midnight the piano on a federal photographer and were out of sorts and social kingdoms your book s have'

In [ ]:
```
seed_text = "I walked out of the store dissatisfied and it"
generate_story(bi_di_gru, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[33]: 'had been devoured grave he cried that he had lost within the character which was three years he could hear johnny called up an d with tears and then he dozed offone day when it was like a midshipman and walked in the island of the fort and a piece of'

In [ ]:
```
seed_text = "The lady was known as a soothsayer - a psychic of sorts. The rubbing of her crystal ball was finalized as she c
generate_story(bi_di_gru, tokenizer, seq_length, seed_text, 50) #generate next 50 words
```

Out[34]: 'on every year and that afternoon or until no doubt leapt from my own determination for off de american boy held his champagne he must cook it hundreds of times but i did not know anything he announced gleefully then remembering the family insisted that he and admiral remain behindit'

In [ ]:
```
seed_text = "Little Chandler's thoughts ever since lunch-time had been of his meeting with Gallaher, of Gallaher's invitatio
generate_story(bi_di_gru, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

Out[35]: 'of course we were s means in melancholy in deep and it turned the cliff and his hair was still more as though it were the elde r by a matter of minutes and then they were conscious of a small family but no nightlight to make a blessed breach no i ll get up he said somewhat all right we re feeling whom he wondered if he had ever been remembered of course i spoke in a muffled whis per and no doubt flemming these stories oh henry daw was free to see he said he thought malignantly i thought about'

In [ ]:
```
seed_text = "On the morning of the fifth of September, in uniform, his revolver on the table, the colonel addressed his soli
generate_story(bi_di_gru, tokenizer, seq_length, seed_text, 100) #generate next 100 words
```

Out[36]: 'stand quite still more than a baron she proffered no explanation his silence was irritating and made her own plans and myself are haunting me for now that all the first time about whose ferocity was rendered gentle which it was late to be he did not go home in the drama that one could derive true enjoyment and become cultivated and humane but do you suppose the public understan ds that she used to say what they want to know what i was asked he remarked somewhat later ef i was asked expectations i though t he d throw over some'

Results:

Deep learning is a branch of machine learning that uses a algorithm that works like a neural network in the brain, and is a with a model called an artificial neural network. These networks are based on a collection of connected units called artificial neurons or neurons. Each connection between neurons can carry a signal from one neuron to another. The receiving neuron processes the signal and sends it to the downstream neurons connected to it. Neurons are organized in layers. Different layers can perform different types of conversions on inputs. The signal passes through the first layer, called the input layer, and reaches the last layer, called the output layer. All layers between these two layers are called hidden layers. Each connection between neurons is assigned an arbitrary weight from 0 to 1. For each layer, the weighted sum of connections pointing to the same neuron in the next layer is calculated. This sum is then passed to the activation function , which converts the output to a number from 0 to 1. The activation function resembles the activity of the human brain,

where different neurons are activated by different stimuli. The conversion result obtained from the activation function is passed to the next neuron in the next layer. This process repeats until the output layer is reached. During training, any weights assigned to each connection between neurons are continuously updated. Create a unique word dictionary from the input file Map words by index Set the length of the sequence Create another tensor (dataY) containing the following words in the sequence in the input file Splits the input file into Tensor (dataX) based on the length of the sequence Hot coding is used to transform the category features into a format that works better with machine learning algorithms. Use the softmax function (activation), where is the activation function, to assign probabilities to the content of dataY. Sets the window size equal to the length of the sequence. Select the next word based on the probability of dataY Repeat this process until you have the required number of words in the output history. Moving the window one word will try to reach the optimized value for the next word, based on the probability of. The optimization of the value depends on the optimizer used. One of the

widely used optimizers is called Stochastic Gradient Descent (SGD). The goal of SGD is to minimize the loss function . The loss function is the difference between the actual output and the output

generated by the model. A single path of all datathat passes through the model is called a single epoch. Model learns through multiple epochs and attempts to minimize the loss function. The model is trained by passing the data in batches. Batches are also commonly referred to as mini-batch. batch size refers to the number of data samples that runs simultaneously over the network. deep learning can take the form of supervised learning when using tagged and unsupervised learning, and supervised learning when using unlabeled data. The Story Scrambler system uses unsupervised learning with a recurrent neural network. Training set of about 30 stories. Each story consists of approximately 5000 words and is processed by the i5 processor system with 4GB of RAM. A tuple of training sets consists of stories divided into strings , the number of words equal to the length of the specified

sequence. Draw loss is calculated by varying different parameters such as RNN size, number of layers, batch size, sequence length.

Conclusion:

Describes the implementation of the Story generator system using RNN's such as GRU's and LSTM's. We tried to minimize drag loss by increasing the values of various parameters such as the number of neurons, the number of layers, the stack size, and the length of the sequence. The formed story is also evaluated by humans, achieving 79% accuracy. The accuracy of system can be further improved by taking into account the meaning of in the context of the word. In addition, synonyms can be used to further improve the accuracy of the system . This system can be further extended for the automatic generation of news or news articles or jokes or posts