Statistical Methods in Artificial Intelligence

Lyric Generation using Recurrent Neural Network

Submitted By:	*Roll No.*
Praveen Balireddy	2018201052
Aman Joshi	2018201097
Arpit Gupta	2018201048
Shubham Pokhriyal	2018201080

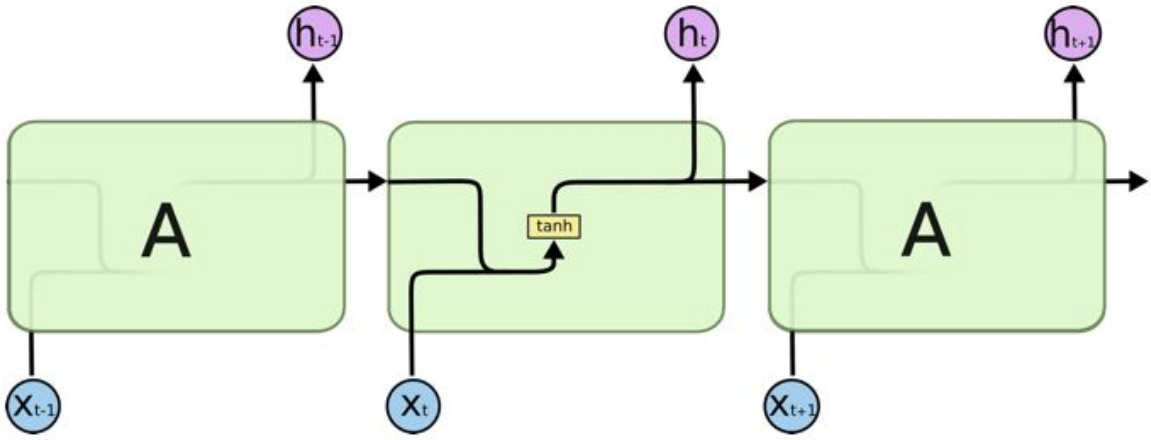
ABSTRACT

This project demonstrates the effectiveness of Recurrent Neural Networks in an efforts to generate song lyrics. The goal of this project is to try out different model architectures and understand how recurrent Neural networks are able to learn the song lyric structures.
The experiments were conducted on an English dataset and on a Hindi to English translated lyric versions.

OVERVIEW

RNN

A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed graph along a sequence. This allows it to exhibit dynamic temporal behavior for a time sequence. Unlike feedforward neural networks, RNNs can use their internal state (memory) to process sequences of inputs. This makes them applicable to tasks such as unsegmented, connected handwriting recognition or speech recognition.
Recurrent Neural Network comes into the picture when any model needs context to be able to provide the output based on the input. Sometimes the context is the single most important thing for the model to predict the most appropriate output.



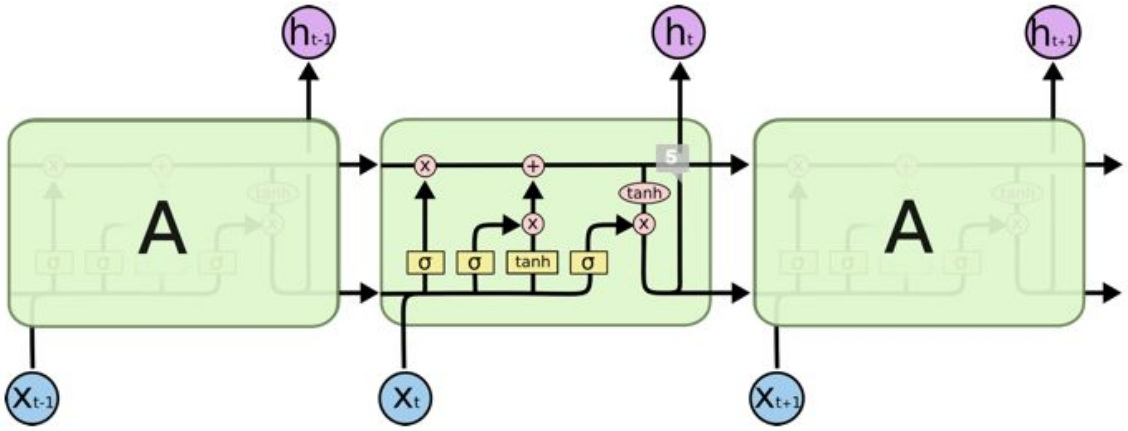
The repeating module in a standard RNN contains a single layer.

LSTM Networks

Long Short Term Memory networks – usually just called “LSTMs” – are a special kind of RNN, capable of learning long-term dependencies. LSTMs are explicitly designed to avoid the long-term dependency problem. Remembering information for long periods of time is practically their default behavior, not something they struggle to learn!

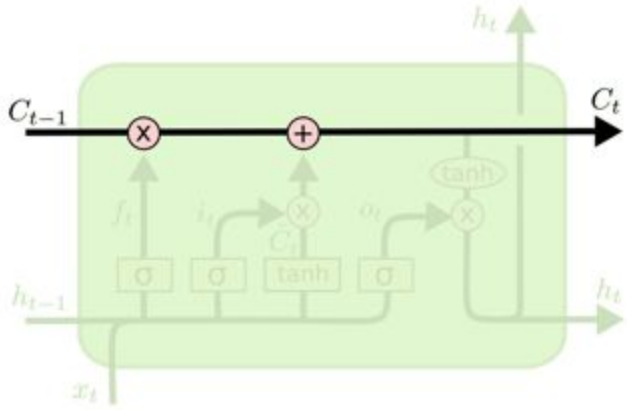
All recurrent neural networks have the form of a chain of repeating modules of neural network. In standard RNNs, this repeating module will have a very simple structure, such as a single tanh layer.

LSTMs also have this chain-like structure, but the repeating module has a different structure. Instead of having a single neural network layer, there are four, interacting in a very special way.



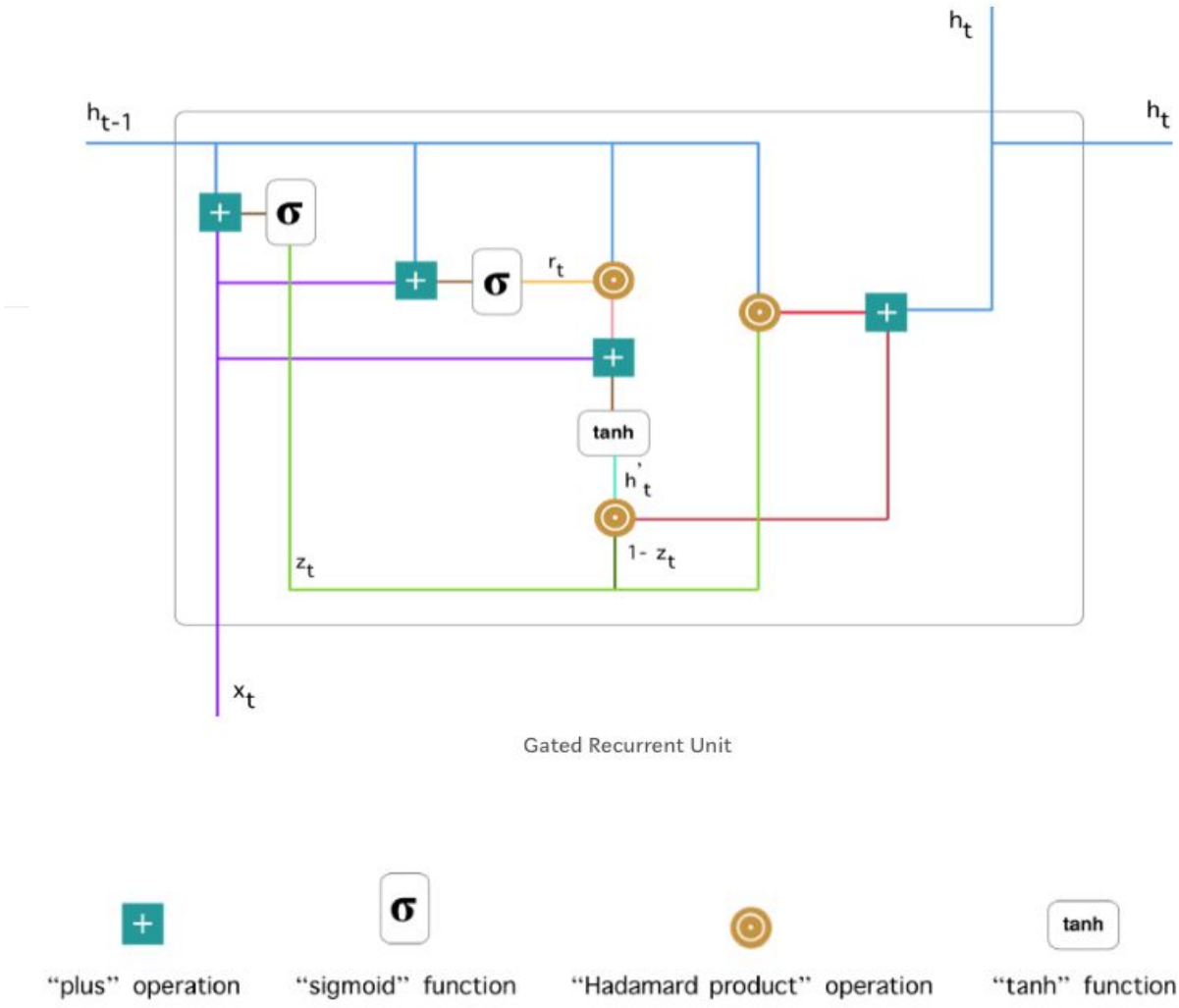
The repeating module in an LSTM contains four interacting layers.

Core Idea behind LSTM



The key to LSTMs is the cell state. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged. The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates. Gates are a way to optionally let information through. They are composed out of a sigmoid neural net layer and a pointwise multiplication operation. The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through!" An LSTM has three of these gates, to protect and control the cell state.

GRU



GRU (Gated Recurrent Unit) aims to solve the vanishing gradient problem which comes with a standard recurrent neural network. GRU can also be considered as a variation on the LSTM because both are designed similarly and, in some cases, produce equally excellent results.

To solve the vanishing gradient problem of a standard RNN, GRU uses, so called, update gate and reset gate. Basically, these are two vectors which decide what information should be passed to the output. The special thing about them is that they can be trained to keep information from long ago, without washing it through time or remove information which is irrelevant to the prediction.

APPROACHES

Word Level Song generator:

The idea here is to train the model with many sequences of words and the target *next_word*. As a simplified example, if each sentence is a list of five words, then the target is a list of only one element, indicating which is the following word in the original text.

```
In [4]: sentences[0]
Out[4]: ['hey', 'there', 'delilah', 'whats', 'it', 'like', 'in', 'new', 'york', 'city']

In [5]: next_words[0]
Out[5]: 'I'

In [6]: sentences[1]
Out[6]: ['there', 'delilah', 'whats', 'it', 'like', 'in', 'new', 'york', 'city', 'I']

In [7]: next_words[1]
Out[7]: 'am'
```

We don't actually send the strings, but a vectorized representation of the word inside a dictionary of possible words (described in the later section). The idea is that after many epochs the model will learn "*the style*" of how the corpus is written, trying to adjust the weights of the network to predict the next word given a sequence of the N previous words.

The text corpus

Corpus is created by concatenating all the songs. The data was noisy and unstructured. Some of the problems with the dataset:

- Whenever a line is going to be repeated twice, it is represented as ----2. This information is not very useful for our model. So we've removed such repetitions. It helps us in achieving our goal "To learn the style of how the corpus is written".
- Preprocessing steps like removing extra spaces, smileys, invalid punctuations like '@', '#', e.tc.
- Cleaning words with one-sided parentheses only.
- Lastly changing each '\n' to ' \n '. Because of this step, we can treat newline character as a separate word. We are leaving the decision of starting new line to model itself.

Getting word frequencies

In the character level text generators, we have very fewer numbers of dimensions one for each character. However, in a word level generator, we have a dimension for each one of the different words, which can turn out to be in the tens of thousands (especially in an unstructured and noisy corpus as dirty as this one).

In order to avoid these large numbers of dimensions, we calculate the frequency of each one of the words, so we can use this information to filter out the uncommon words, thus reducing the dimensionality of the data. Hence the memory and time to train the network significantly reduced. We have filtered out the words which have a frequency less than MIN_WORD_FREQUENCY. This is done to 'make the cut' and select only the frequent words to form the final dictionary of the words. We also created the dictionaries to translate from word to index and from index to word.

Creating and filtering the sequences

We at this point we have *text_in_words* which is an array containing all the corpus word-by-word. We created sequences of size *SEQUENCE_LEN* (another parameter that can be picked by hand) and store them in *sentences*, and in the same index, store the next word in *next_words.*

```
sentences = []
next_words = []
ignored = 0
for i in range(0, len(corpus) - maxlen):
    # Only add the sequences where no word is in ignored_words
    if len(set(corpus[i: i+maxlen+1]).intersection(ignored_words)) == 0:
        sentences.append(corpus[i: i + maxlen])
        next_words.append(corpus[i + maxlen])
    else:
        ignored = ignored + 1
print('Ignored sequences:', ignored)
print('Remaining sequences:', len(sentences))
```

But there is a problem: in *text_in_words* we still have many of the words to be ignored. We cannot just go ahead and remove these words because we would be breaking the language and leaving incoherent sentences. That's why we need to validate each possible sequence+next_word, it should be ignored it if contains at least one of the ignored words. Without word filtering, we would've been ending up using a large number of parameters requiring 100s of GB of memory.

Hence we used data generators. Using data generators, you feed the model with *chunks* of the training set, one for each step, instead of feeding everything at once. The generator function gets a list of sentences and next_words, and the size of the batch. Then it yields two numpy arrays of *batch_size* consisting of input sentence and output words. We use the *index* variable to keep track of the examples we have already returned. It is re-initialized to 0 whenever we reach the

end of the lists. This generator can be used for both training and evaluation (just passing a different *sentence_list* and *next_word_list*).

```
def generator(sentence_list, next_word_list, batch_size):
    '''
    Generator function to generate the input/output data using
    generators concept(to avoid RAM overflow)
    '''
    index = 0
    while True:
        x = np.zeros((batch_size, maxlen, vocab_size), dtype=np.bool)
        y = np.zeros((batch_size, vocab_size), dtype=np.bool)
        for i in range(batch_size):
            for t, w in enumerate(sentence_list[index]):
                x[i, t, char_ix[w]] = 1
            y[i, char_ix[next_word_list[index]]] = 1

            index = index + 1
            if index == len(sentence_list):
                index = 0
        yield x, y
```

After all these steps data is fed to model for training

Character level Song Generator

The idea here is to train the model with many sequences of characters and the target *next_character*. A simplified example is shown below.

```
In [18]: sentences[0]
Out[18]: ['h', 'e', 'y', ' ', 't', 'h', 'e', 'r', 'e', ' ']

In [19]: next_char[0]
Out[19]: 'd'

In [20]: sentences[1]
Out[20]: ['e', 'y', ' ', 't', 'h', 'e', 'r', 'e', ' ', 'd']

In [21]: next_char[1]
Out[21]: 'e'
```

We directly send *Maxlen* characters to the model. The idea is that after many epochs the model will learn “the style” of how the corpus is written, trying to adjust weights of the network such that the model will be able to predict the next character given a sequence of *Maxlen* previous characters.

The text Corpus

The corpus creation was more or less the same as that of the word based model. However, there were few differences as follow:

- The space is treated as part of input here. While in word-based model space is used to split the sentence into words, here it is treated the same as that of any other character.
- The conversion of ‘\n’ to ‘ \n ’ is not performed here because we are treating space also as a part of input for model.

Creating Sequence

We created sequences of *Maxlen* (hyperparameter) and store them in *sentences* as in the same index store next character in the *next_char*.

```
sentences=[]
next_char=[]
for i in range(len(corpus)-maxlen-1):
    sentences.append(corpus[i:i+maxlen])
    next_char.append(corpus[i+maxlen])
split_count = int(0.8 * len(corpus))
sentences_test = sentences[split_count:]
next_char_test = next_char[split_count:]
sentences = sentences[:split_count]
next_char = next_char[:split_count]
```

Then we used generator same as described in word-based approach to feed data to model.

EXPERIMENT AND RESULTS

Following results are obtained by training different architectures. Various parameters like - Number of LSTM layers, epochs, timesteps, number of neurons in hidden layer, etc. were tweaked to get different architectures.

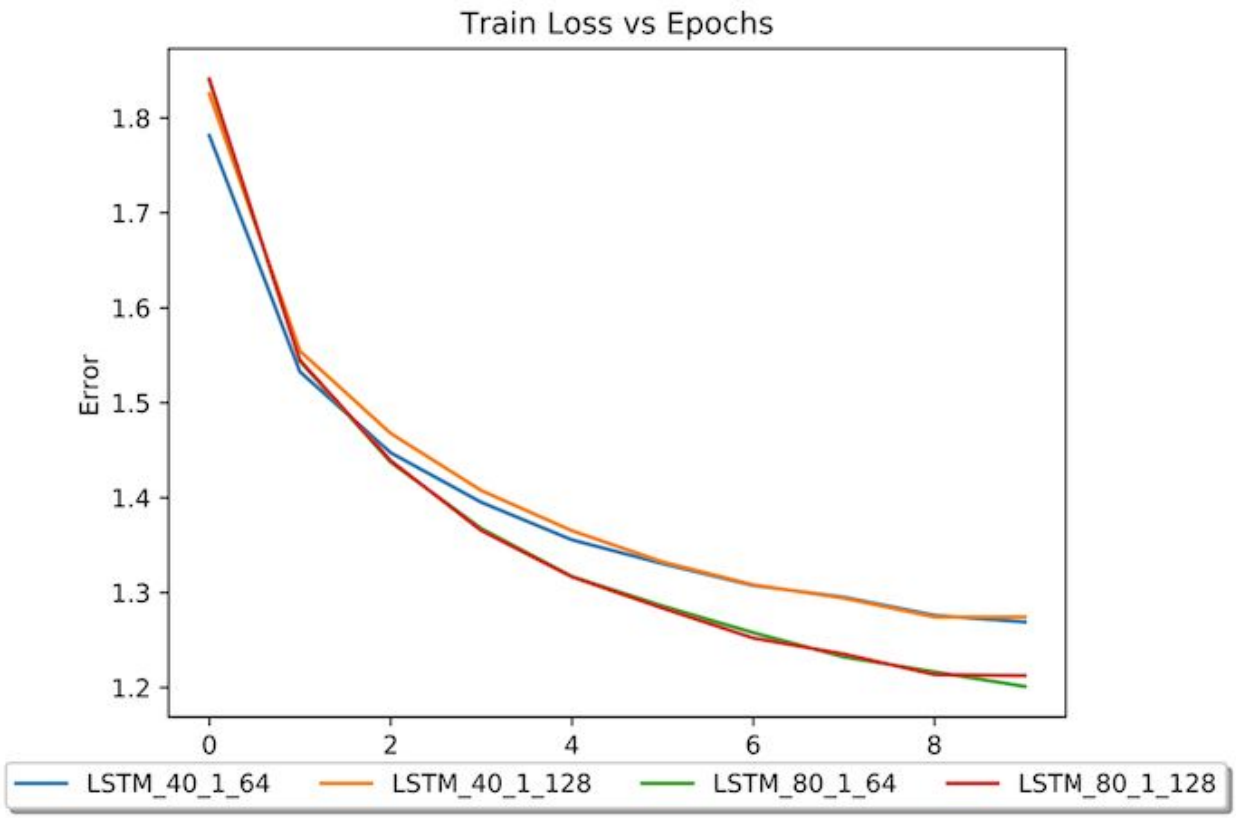
In the following tables first column represents the architecture used. It can be read as - (Model-Timesteps-Layers-Neurons). For example, LSTM_40_1_64 means single layer LSTM model with 64 neurons in hidden layer was used where timesteps were 40.

English songs dataset

Char-based Approach

Architecture	Input (seed)	Generated lyrics
LSTM_40_1_64	“ work this treacherous road for all the “	work this treacherous road for all the side is that some could i don't know on was and but of your down and when i thought the light of i a
LSTM_40_1_128	“the chance on you chorus the first noel “	the chance on you chorus the first noel the sayin' for a little boy and i can't follow baby i was some to my search that i say i was long an
LSTM_40_2_128	“ the mother mother touched, and dude ain”	the mother mother touched, and dude ain't down to hear a look is so bad with you so dead to help and we know that i ride my read or on our
LSTM_80_1_64	"yes i know so, there's other fish in	yes i know so, there's other fish in the sea who would love to swim

	the sea who would love to swim with me you'"	with me you're so on the sound of your song i'll be don't little great i wish the sun you how the moon you last
LSTM_80_1_128	"early the books, the paintings and the furniture help me the signal's sounding o”	early the books, the paintings and the furniture help me the signal's sounding out a little alone gettin' everybody the sun like the sands then the mountain i've done when you so l

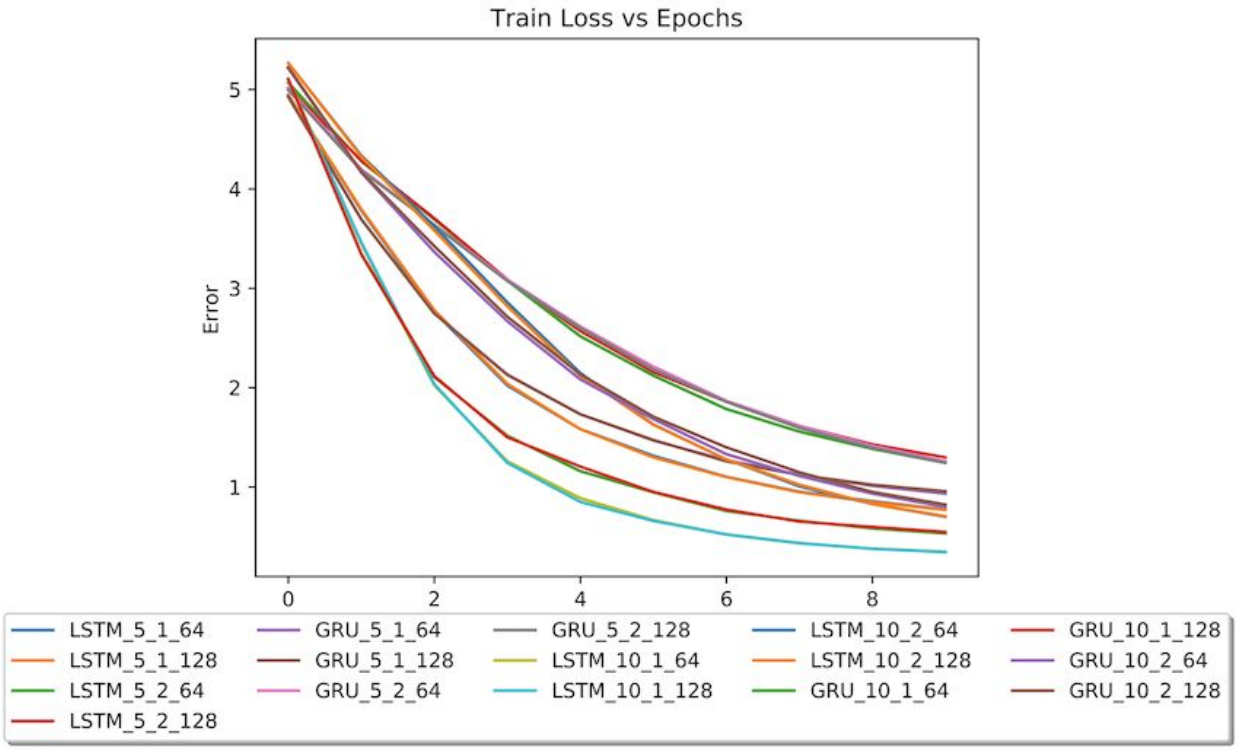


Word-based Approach

Architecture	Input (seed)	Generated lyrics
GRU_5_1_64	"Sand walking with someone”	sand walking with someone and holding his hand i should walk with me it seems i stood and forget all sometimes lovers don't do all they could sometimes lovers hide in the cover of innocence and blame i know that we act just the same your heart's
GRU_5_2_128	"stop the tears how"	stop the tears how i hate to spend our dust in the stars eyes for the end of the road so, look out for me, darling on the power of love a man don't crawl that we need to walk around here tonight gotta go down
GRU_10_1_64	"you made me come alive [chorus] i'll"	you made me come alive [chorus] i'll never come home and i think i'll get my love 'cause i love you. and i know i was never in i can't take it from you

		i know it's not worth trying so long, so long, so long so long,
GRU_10_2_128	"greatest times i've had when i was dancing with you"	greatest times i've had when i was dancing with you i can't feel like your best man can i've had way too much tonight please don't tell me no more i don't care maybe i don't want to go outside out i couldn't drown it but as much as tomorrow
LSTM_5_1_64	"'cause i'm in love with"	'cause i'm in love with a beautiful gal but she don't care about me and i closed my eyes, will leave me girl then i don't want to break these chains until we want to be free this is the part when i break free cause i
LSTM_5_2_128	"i know that can never"	i know that can never end the love will be a better way may each day in the year be the one i never wanna learn gimme gimme gimme a man after midnight won't you be alone and you remember in the hills of mexico well the

LSTM_10_1_64	"he loves you and life has just begun it's"	he loves you and life has just begun it's the time more than any words can say and i am just a girl not the kind of woman men would like to meet just another girl no one ever looks at in the street but today i can't believe it's true when you
LSTM_10_2_128	"it all right you're all i ever need, my"	it all right you're all i ever need, my darling and i would just go no to give you this heart of mine you know what i need when i try to explain it i be that's as i sit here today many starts is on tonight how i need a

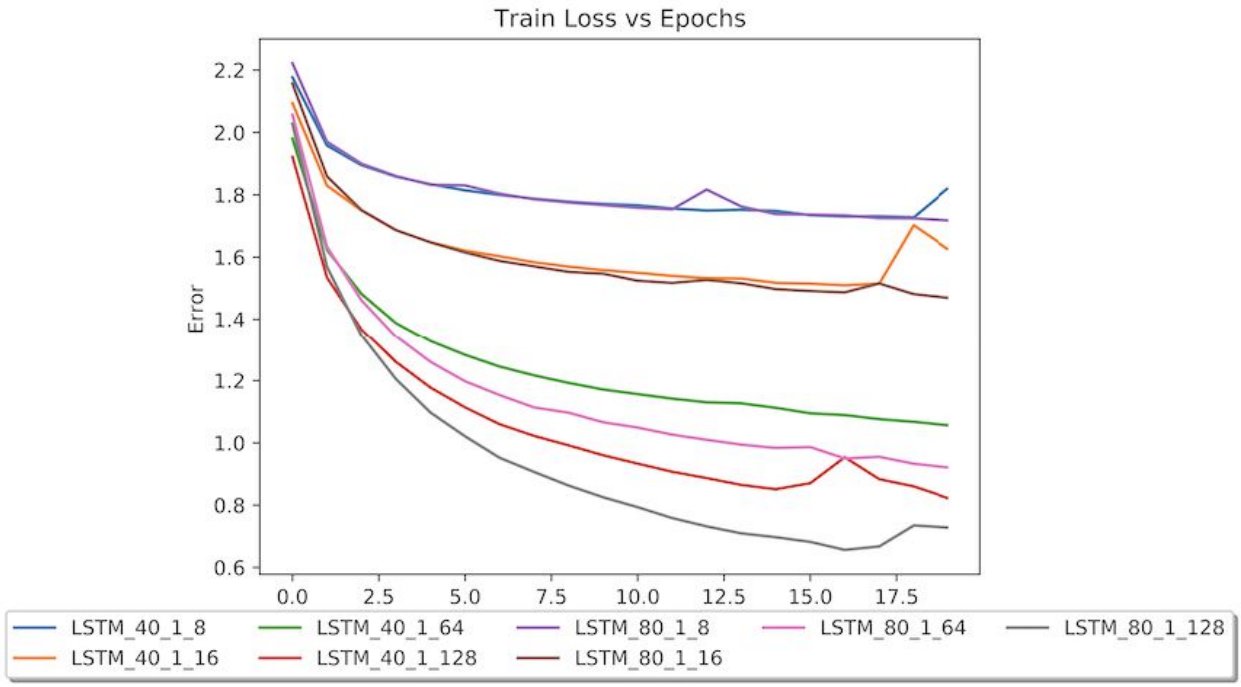


Hindi songs translated dataset

Char-based Approach

Architecture	Input (seed)	Generated lyrics
LSTM_40_1_8	"hout you chorus i wont live i will die w"	hout you chorus i wont live i will die wanc in as in and beauth a my heart in beant like me the what no le the san and in sore and in and in
LSTM_40_1_16	"eloved didnt come beloved didnt come tha"	eloved didnt come beloved didnt come that beloved didnt come beaut the beade god on shall and shall god about you are and seare you are the
LSTM_40_1_128	"ont know i dont know o heart be steady y"	ont know i dont know o heart be steady youre how destinations brought in my prayer that god now now my eyes yeed in meeting even all curl cr
LSTM_80_1_8	" someone let loose my heart and filled it with love just with his eyes he spoke "	someone let loose my heart and filled it with love just with his eyes he spoke to my fros am thise the awe the eoy the sometimes or and you sous is of one tole of tire not the you
LSTM_80_1_16	"bless waist doesnt listen to me when there is a beat playing i gotta shake it li"	bless waist doesnt listen to me when there is a beat playing i gotta shake it like the world of hor so so in my heart is ange this in on the beauting in the my someone in the songo

LSTM_80_1_128	"p of your and mine it doesnt have a face and yet its so deep these moments these"	p of your and mine it doesnt have a face and yet its so deep these moments these as your say fared its see the say you are the say yes



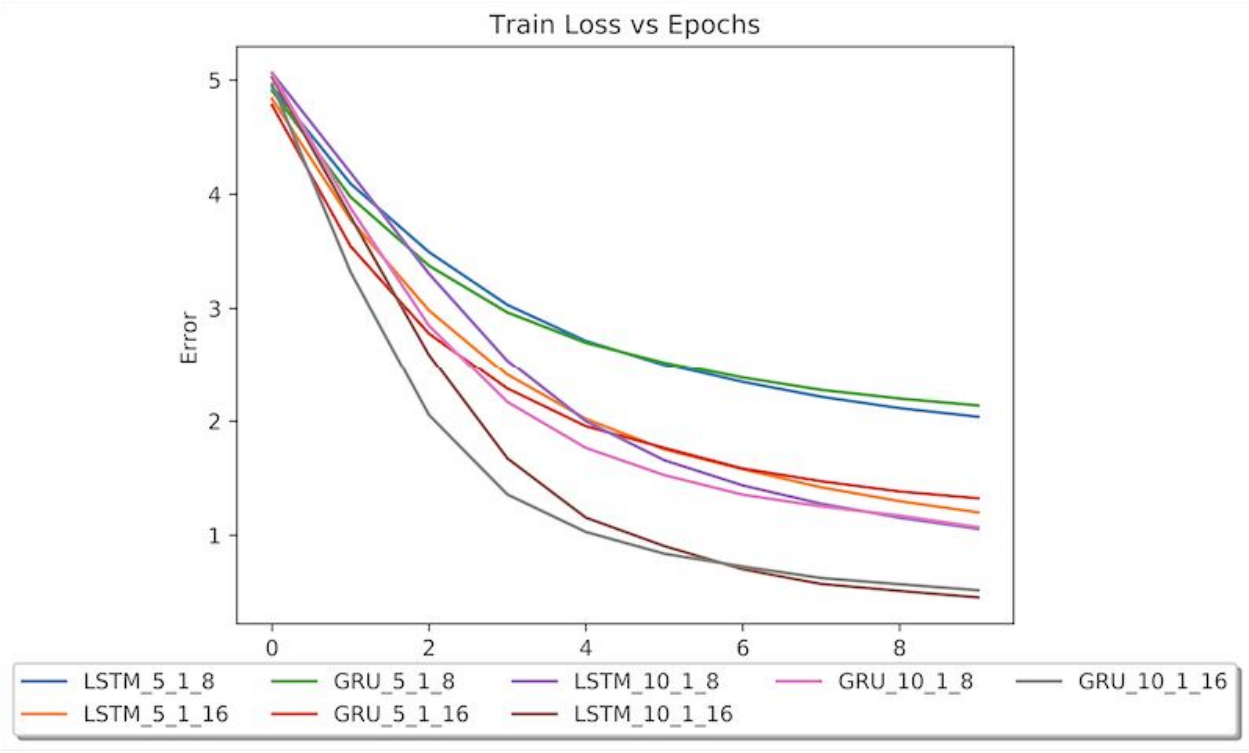
Word-based Approach

Architecture	**Input (seed)**	**Generated lyrics**
GRU_5_1_8	"oh eyes deceivers "	oh eyes deceivers your eyes are big deceivers you are the days of the night you are in my eyes ill take care of you in my breaths ill keep you in my eyes

		ill take care of you in my breaths ill keep
GRU_5_1_16	" shake that "	 shake that youll he became full of joy since you are the bride of badri baby see the girls you burns of live without you dont talk about the world now the world of stars now since i have seen thy face i have
GRU_10_1_16	"are set just for you i have forgotten the"	are set just for you i have forgotten the whole world just for you ooo i am u i am love you are so beautiful i could die to see you give me some so is incomplete i am your shadow all the same and yet is

LSTM_5_1_8	"o o o if"	o o o if thou cry you are there for you you are my prayer you are mine you are mine you are you beautiful how do i tell you you are my desire you are a dream in the world
LSTM_5_1_16	"you have the way of"	you have the way of your eyes ill keep you in my eyes ill take care of you in my breaths you are my life is lost that you are in front of me i am very crazy i agree that i have to myself
LSTM_10_1_8	"world seems lonely the world seems lonely "	world seems lonely the world seems lonely when no loved one remains the world seems lonely the world seems lonely when no loved one remains the world seems lonely the bride seems lonely when this is not we became your heart

		what story your name these one ooo
LSTM_10_1_16	"for a moment i want to become your bride"	for a moment i want to become your bride of the was of the girls became the story brought the love is a very first decorate the world the seems as beloved we as well take you away the bride of badri well take you away making you



Team Members and Tasks assigned

1. Praveen Balireddy:
    a. Built POC for char based approach
    b. Experimented with some of the architectures to see the impact of hyperparameters
    c. Set up VM on google cloud
    d. Prepare github readme
2. Aman Joshi:
    a. Wrote code to generate results for different architectures
    b. Explored on the keras open issue of random predictions after loading model
    c. Contributed to report making
3. Arpit Gupta:
    a. Built POC for word based approaches
    b. Data Scraping and Dataset creation for english/hindi-english translated songs
    c. Created flask based web application
    d. Contributed to report making
4. Shubham Pokhriyal:
    a. Explored different approaches from research papers
    b. Prepared presentation
    c. Contributed to report making

Reference

1. https://github.com/keras-team/keras/blob/master/examples/lstm_text_generation.py
2. https://towardsdatascience.com/ai-generates-taylor-swifts-song-lyrics-6fd92a03ef7e
3. https://medium.com/coinmonks/word-level-lstm-text-generator-creating-automatic-song-lyrics-with-neural-networks-b8a1617104fb
4. http://www.emnlp2015.org/proceedings/EMNLP/pdf/EMNLP221.pdf
5. https://nlp.stanford.edu/courses/cs224n/2009/fp/5.pdf