Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 143 (2018) 117–123

Procedia Computer Science

www.elsevier.com/locate/procedia

8th International Conference on Advances in Computing and Communication (ICACC-2018)

A RNN based Approach for next word prediction in Assamese Phonetic Transcription

Partha Pratim Barman[a], Abhijit Boruah[a]

[a]*Dept of CSE, DUIET, Dibrugarh University, Dibrugarh-786004, Assam, India*

Abstract

In this paper, we present a Long Short Term Memory network (LSTM) model which is a special kind of Recurrent Neural Network(RNN) for instant messaging, where the goal is to predict next word(s) given a set of current words to the user. This method is more complex in other languages apart from English. For instance, in Assamese language, there are some equivalent synonyms of the word 'you', that is used to address a second person in English. Here, we have developed a solution to this issue by storing the transcripted Assamese language according to International Phonetic Association(IPA) chart and have fed the data into our model. Our model goes through the data set of the transcripted Assamese words and predicts the next word using LSTM with an accuracy of 88.20% for Assamese text and 72.10% for phonetically transcripted Assamese language . This model can be used in predicting next word of Assamese language , especially at the time of phonetic typing.

Keywords: Next word prediction; LSTM ; RNN; Assamese.

1. Introduction

Electronic conversation and communication among masses is a common phenomenon in today's world of real-time social media. Most of the time, the conversations use a transcripted form of a native language(other than English) for informal conversations. By reducing the time consumption for typing, the next word prediction in transcripted texts would be very helpful for day to day usage and ease of communication. Local Indian languages like Assamese is exposed to a more personalized level of such communication. For example, there are three different words for denoting a person: *tumi, toi* and *apuni* in Assamese. Based on these three types of addressing, for a single sentence in English, there can be three or more possible sentences in Assamese.

* Corresponding author. Tel.: +91-7896501390 ; fax: +0-000-000-0000.
E-mail address: ppbarman18@gmail.com

10.1016/j.procs.2018.10.359

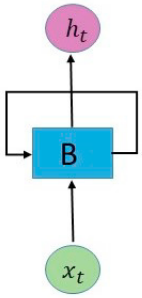

Fig. 1. A NN with some input value x_t and outputs a value h_t [7]

While prediction of the next word in any regional language, the main problem is that some of our machines understand only ASCII values which is the most common format of text files in the computer and the Internet in which each alphabetic, numeric, or special character is represented with a 7-bit binary number. When we use Unicode for the Assamese language, some of the characters of Assamese is not printed properly on the screen while implementing on platforms like python. So we have phonetically transcripted a collection of Assamese words using Google Indic keyboard and stored them in a text file which we will use for training our model. We will also use the original Assamese words to train the model, check the accuracy and compare it with the prediction performance while using transcripted Assamese words. The rest of the paper is organized as follows. Prerequisites for this work including basics of neural networks like RNN and LSTM are discussed in section 2. Section 3 illustrates our proposed method for word prediction including preparation of Assamese transcripted and non-transcripted dataset. A discussion of the experimental results is presented in section 4 along with the conclusion and future directions of work in section 5.

2. Background

Next word prediction is a highly discussed topic in current domain of Natural Language Processing research. Felix et. al (1999) [3] used LSTM to solve tasks that were previously unsolvable by RNNs. They introduced forget gates to solve continual versions of these problems. Mikolov et. al(2010) [6] presented a simple Recurrent Neural Network based language model to improve prediction of next word in sequential data. In another work, Alex(2013)[4] discussed the use of LSTM to generate complex long-range structured sequences. The author implemented the method for text and online handwriting prediction with an extension to handwriting synthesis. Sukhbataar et. al (2015) [8] introduced a Recurrent Neural Network to perform next word prediction on a text sequence. Their proposed model was an extension of RNNSearch [1] and the model was trained by backpropagation on diverse tasks from question answering to language modeling. Recently, Zhou et. al (2015) [9] proposed a novel hybrid method called C-LSTM, where text classification and predictions on sentence representations were implemented by using a combined approach of RNN and LSTM. In this section, we discuss in brief two necessary topics LSTM and RNN, that are essential to our proposed model.

2.1. Recurrent Neural Networks

Human beings don't ponder over new thoughts on their own every second. While going through a text, we acknowledge each word based on our understanding of the previous words. We have the capacity to relate and our thoughts are continuous. Traditional neural networks fall short during such perceptions, which is a major drawback. For example, imagine we want to classify what kind of event is happening at every point in a novel. It's unclear how a traditional neural network could use its reasoning about previous events in the storyline to inform later ones. Recurrent neural networks can be used to address this issue. In Figure 1, neural network B have some input value x_t and outputs a value h_t. A loop allows information to be passed from one step of the network to the next. Figure 2 is a unrolled

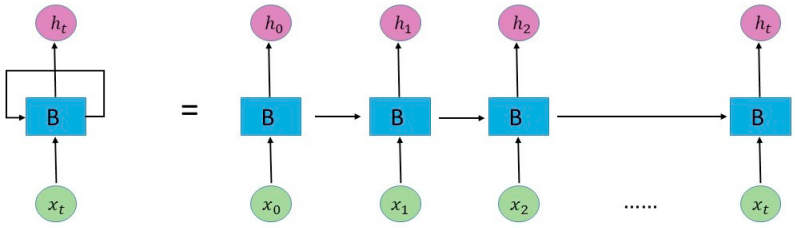

Fig. 2. Unrolled recurrent neural network for NN Figure 1. [7]

recurrent neural network where at times, only recent information is required to perform the present task. For instance, let us examine a language model trying to predict the next word on the basis of the previous ones. If we are trying to predict the last word in *'The sun rises in the east'*, we don't require any other conditions as east being the pretty obvious next word in this case. In these cases, where the gap between the relevant information and the place that it's needed is small, RNNs can learn to use the past information.

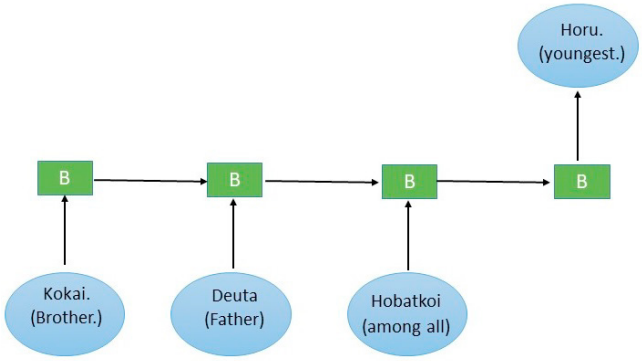

Fig. 3. An example scenario of the implementation.

But there are also cases where we require more conditions. Consider trying to predict the last word in the text *'I am good at playing cricket and recently I represented my state in cricket.'* Recent information suggests that the next word is probably the name of a sport, but if we want to narrow down which sport, we need the information about cricket, from further back. Here the gap between the relevant information and the point where it is needed becomes very large. Unfortunately, as that gap grows, RNNs become unable to learn to connect the information. The problem was explored in depth by Bengio et.al (1994)[2] who found some pretty fundamental reasons why it might be difficult.

Figure 3 describes an implementation scenario using the LSTM network as discussed in Figure 2. If we input three consecutive words from a sentence (*'Kokai'*, meaning brother, *'Deuta'*, meaning father and *'hobatkoi'*, meaning amongst all), to the LSTM network, the predicted output is *'horu'* (meaning youngest), which is a correct prediction according to the dataset.

2.2. LSTM Networks

Long Short-Term Memory networks are a special kind of RNN, capable of learning long-term dependencies. They were introduced by Hochreiter & Schmidhuber (1997)[5]. 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 main key to LSTM is the cell state: it runs straight to the entire chain, with some minor linear interactions. It is very easy in it for information to just flow along unchanged. Gates allow LSTM to add or remove information to the cell state. They are composed of a sigmoid neural net layer and a pointwise multiplication operation. LSTM has three of these gates, to protect and control the cell state. The first step in our LSTM is to decide what information were going to throw away from the cell state. This decision is made by a sigmoid layer called the forget gate layer. It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. An output of 1 represents *'completely keep this'* while a 0 represents *'completely get rid of this'*. Here h_{t-1} is the output of the previous LSTM block, x_t is the input and h_t is the output of the current LSTM block, C_{t-1} is the cell state output from previous LSTM block, W_f is the weight vector and b_f is the bias of forget gate layer. f_t is the output of the forget gate layer.

$$f_t = \sigma(W_f.[h_{t-1}, x_t] + b_f) \tag{1}$$

The next step is to decide what new information were going to store in the cell state. This has two parts: first, a sigmoid layer called the input gate layer decides which values we will update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, well combine these two to create an update to the state. Here i_t is the output from the input gate layer, W_i is the weight vector and b_i is the bias of input gate layer. W_C is the weight vector and b_C is the bias for creating a vector of new candidate values, $\tilde{C}_t$. C_t is the updated cell state.

$$i_t = \sigma(W_i.[h_{t-1}, x_t] + b_i) \tag{2}$$

$$\tilde{C}_t = tanh(W_C.[h_{t-1}, x_t] + b_C) \tag{3}$$

We multiply the old state by h_{t-1} , forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{4}$$

Finally, we need to decide what were going to output. This output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state were going to output. Then, we put the cell state through tanh (to push the values to be between -1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to. Here o_t is the output from the output gate layer, W_o is the weight vector and b_o is the bias of the output gate layer. h_t is the output of the current LSTM block.

$$o_t = \sigma(W_o.[h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t * tanh(C_t) \tag{6}$$

3. Proposed Method

For our RNN model we have prepared the dataset from an Assamese novel named **Bhanumati** written by Assamese scholar *Padmanath Gohainbaruah*. We have created two types of data-set. The first data-set contains first 1037 words of the novel Bhanumati and these words are repeated for five times to increase the number of words in the data-set. The second data-set is phonetic transcription of the first data-set word by word till the end and contains 1076 words. We can see that there is an increase in number of words in the second data-set as the number of words increase when we phonetically transcript them. These two data-set are fed into our RNN models and results are calculated at after every 1000 epochs. Since displaying the results of every 1000 epoch will be a large table, we have presented in Table 1 the epoch at which the maximum average accuracy is achieved among the calculated ones.

For testing our model we are feeding three consecutive random words to our model and predicting the immediate next word. The three consecutive words for testing must be random consecutive words from the training set as our LSTM network must be familiar to words which it is going to predict. The duration for training up to 100000 epochs takes 5 minutes for the current dataset.

Table 1. Structure of different models with it's highest average accuracy with its epoch for Assamese language.

Test number	Hidden Layers	Neurons in each hidden layer	Learning Rate	Epoch	Average Accuracy
Test 1	2	128	0.001	98000	88.20%
Test 2	2	256	0.001	82000	77.20%
Test 3	2	512	0.001	86000	70.40%
Test 4	3	128	0.001	100000	81.10%
Test 5	3	256	0.001	58000	74.30%
Test 6	3	512	0.001	50000	71.90%
Test 7	4	512	0.001	98000	70.30%
Test 8	4	512	0.003	74000	22.20%

Table 2. Table shows how Assamese words are transcripted to Phonetically transcripted words.

Assamese Words	Phonetically transcripted words	Assamese Words	Phonetically transcripted words
মোৰ	Mur	নাম	naam
ভানুমতী	Bhanumotee	দেউতাৰ	deutar
বহু	bahu	সাধনাৰ	hadhonar
ফল	fol	একেটি	eketi
ৰত্ন	rotno	জীৱনৰ	jibonor

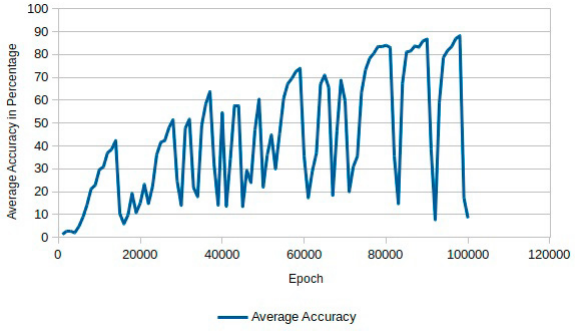

Fig. 4. Graph displays Average Accuracy for Test 1.

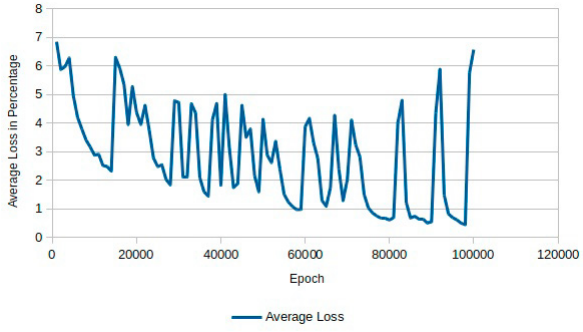

Fig. 5. Graph displays Average Loss for Test 1.

4. Experimental Results

We fed the first dataset(the Assamese language without transcription) into different models with different structure till 100000 epochs. The epochs with maximum accuracy for different models are given in Table 1. This table is for the first data-set and we can see that maximum accuracy is 88.20% when the model has two layers, each layer have 128 neurons,learning rate of 0.001 and 98000 epoch.

Figure 4 displays the average accuracy after every thousand epoch and Figure 5 displays the average loss after every thousand epochs of Test 1, which have maximum accuracy among different model having different configurations when we fed the first data-set.

Similarly, the second data-set which contains phonetically transcripted words is fed to different models with different configurations. Every model is run for 100000 epoch and the epoch in which we get maximum average accuracy

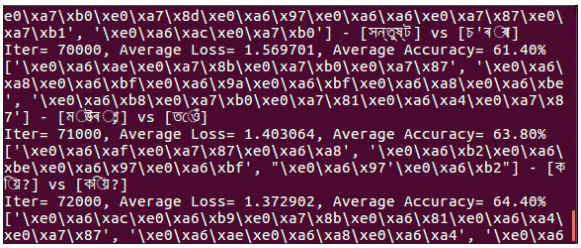

Fig. 6. Figure displays the training model for Assamese language without transcription.

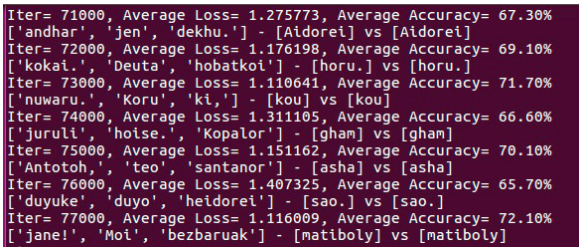

Fig. 7. Figure displays the training model for Assamese language with transcription.

Table 3. Structure of different models with it's highest average accuracy with its epoch for Phonetically Transcripted Assamese language.

Test number	Hidden Layers	Neauron's in each hidden layer	Learning Rate	Epoch	Average Accuracy
Test 9	2	128	0.001	100000	46.10%
Test 10	2	256	0.001	100000	67.00%
Test 11	2	512	0.001	77000	50.60%
Test 12	3	128	0.001	68000	56.40%
Test 13	3	256	0.001	77000	72.10%
Test 14	3	512	0.001	98000	63.60%
Test 15	4	512	0.001	91000	70.40%
Test 16	4	512	0.003	34000	16.10%

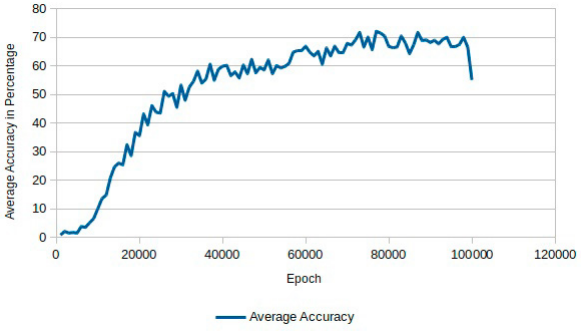

Fig. 8. Graph displays Average Accuracy for Test 13.

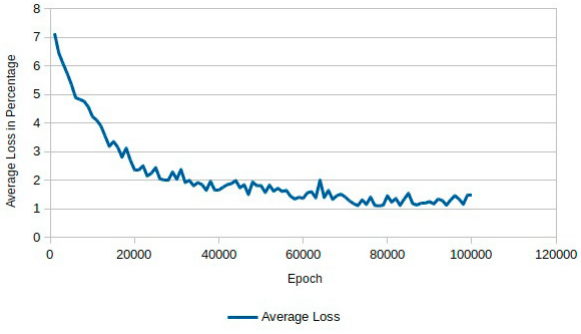

Fig. 9. Graph displays Average Loss for Test 13.

for each model is shown in Table 3. Since displaying the results of every 1000 epoch will be a large table, we have presented in Table 3 the epoch at which the maximum average accuracy is achieved among the calculated ones.

From this table, it is seen that maximum accuracy attained is 72.10% when the model has three layers, where each layer has 256 neurons, learning rate of 0.001 and 77000 epochs. Figure 8 displays the average accuracy after every thousand epoch and Figure 9 displays the average loss after every thousand epochs of Test 13, which have maximum average accuracy among different models. Figure 7 is the actual screenshot while running our model for the transcripted Assamese language. Here it is seen that after every 1000 iteration we are calculating the average loss and average accuracy for our model. We are feeding three consecutive words to our model and predicting the immediate next word. For example, in the Figure 7 , we can see that andhar jen dekhu is the three consecutive word and by feeding this in our model we are predicting the next word Aidorei which is the actually the next word of sentence andhar jen dekhu . Similarly, Figure 6 is the screenshot while running our model for the Assamese language without transcription. We can see that the Assamese words are not displayed in the python environment but they

are represented with some Unicode which is not human readable. Each word in Assamese have some unique code assigned by the computer and with help of this Unicode it tries to predict the next word after feeding these words into the model.

From the above test, we can see that for the Assamese language without transcription in Table 1, the Average accuracy decreases with the increase of number of neurons in each hidden layer by fixing the number of hidden layers. The average accuracy is also effected by the number of hidden layer and learning rate. With the increase of hidden layers, the average accuracy decreases for our model and when we increase the learning rate from 0.001 to 0.003 in Test 8 we can see that there is a drastic decrease in average accuracy. But for phonetically transcripted Assamese language in Table 3 we can see that average accuracy increases when we increase the number of neurons from 128 to 256 in each layer. But the further increase of neurons to 512 in each layer decreases the accuracy. In fact, we can see that with the increase of hidden layers from 2 to 3, there is an increase in the average accuracy but further increase of hidden layers leads to a decrease in the average accuracy. Similarly when we increase the learning rate from 0.001 to 0.003 the average accuracy decreases drastically. From the Figure 4 and Figure 5 which displays the average accuracy and average loss for Test 1, we can see that there exist many local minima point in our model for the Assamese language without transcription. With the increase with the number of epochs, we can see the variations in the average loss and we can see the global minimum at epoch number 98000 in Test 1. Similarly, Figure 8 and Figure 9 displays the average accuracy and average loss for Test 13. We find the global minima of average loss at epoch 77000 in Test 13.

5. Conclusion & future directions

This work towards next word prediction in phonetically transcripted Assamese language using LSTM is presented as a method to analyze and pursue time management in e-communication. We compared the accuracy of our RNN model between Assamese language and Phonetically transcripted Assamese language. From the results, it is seen that first data-set (Assamese text without transcription) have more accuracy compared to the accuracy of the second data-set (Assamese text with transcription).This is due to an increase in the number of words in the second data-set being phonetically transcripted. It will be interesting to observe the results with much bigger datasets than the currently used one. Preparation of a big dataset of phonetically transcripted local Indian languages is a tedious job as no online sources are available for a lot of them and it has to be processed manually. Since a database of phonetically transcripted words in the Assamese language is not available, we tried to increase the number of words by repetition. In our future course of action, we plan to build larger datasets than the currently used one, implement our models and analyze the results. Even though the Assamese language is focused in our work, this LSTM model can be applied to other local languages also.

References

[1] Bahdanau, D., Cho, K., Bengio, Y., 2014. Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473 .

[2] Bengio, Y., Simard, P., Frasconi, P., 1994. Learning long-term dependencies with gradient descent is difficult. IEEE transactions on neural networks 5, 157–166.

[3] Gers, F.A., Schmidhuber, J., Cummins, F., 1999. Learning to forget: Continual prediction with lstm .

[4] Graves, A., 2013. Generating sequences with recurrent neural networks. arXiv preprint arXiv:1308.0850 .

[5] Hochreiter, S., Schmidhuber, J., 1997. Lstm can solve hard long time lag problems, in: Advances in neural information processing systems, pp. 473–479.

[6] Mikolov, T., Karafiát, M., Burget, L., Černockỳ, J., Khudanpur, S., 2010. Recurrent neural network based language model, in: Eleventh Annual Conference of the International Speech Communication Association.

[7] Olah, C., 2015. Understanding lstm networks. URL: http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[8] Sukhbaatar, S., Weston, J., Fergus, R., et al., 2015. End-to-end memory networks, in: Advances in neural information processing systems, pp. 2440–2448.

[9] Zhou, C., Sun, C., Liu, Z., Lau, F., 2015. A c-lstm neural network for text classification. arXiv preprint arXiv:1511.08630 .