Turkish Online Journal of Qualitative Inquiry (TOJQI)
Volume 12, Issue 5, July, 2021: 4284 - 4292

Research Article

Next Word Prediction In Telugu Sentences Using Recurrent Neural Networks

Dr. A. Srinagesh[1], Dr. Ch.Sudha Sree[2], Dr.B.Prasanthi[3], Mr.P.Rama Krishna[4], Mr. P. Siva Prasad[5]

Abstract

Data that is collected from the social network is almost full of noise and extra characters. Popular social networks like Twitter, Facebook, WhatsApp, YouTube generate lots of data and preprocessing is the primary task to transform the messy data into useful or reliable data. The preprocessing step removes, replaces, modifies, and normalizes the collected data. It also fills the data with suitable words in incomplete sentences. In this paper, we address the procedure to predict suitable words in the incomplete sentence in the Telugu language.

Natural Language Toolkit, for example, Indic Language (iNLTK) supports the Indian language for processing 13 official languages in India. Here, Telugu is one of the languages that is supported by iNTK. The prediction of the next word can be implemented by using Recurrent Neural Networks, by predicting the next word in a sequence so that the number of keystrokes of the user can be reduced to save time. This approach can be used effectively for various NLP preprocessing methods, sentence auto-completion in Telugu.

Keywords-- Natural Language Processing, Next Word Prediction, Recurrent Neural Networks, Long Short Term Memory (LSTM).

I. INTRODUCTION

Next Word Prediction [1] in Telugu sentences is the process of predicting the required and suitable words for a messy sentence or missing words in the sentence. In order to normalize the social networks sentences, preprocessing techniques are needed to implement this task. To improve the accuracy of the input data, deep learning techniques can be used. In deep learning, we have used recurrent neural networks to implement the task. Recurrent [6,11] Neural Networks are also used for time series prediction, under recurrent neural networks, we have Long Short Term Memory (LSTM) recurrent neural networks which are used to predict the next word.

In this paper, we build a model by using LSTM [2] and that model gets trained with the Telugu text data and then we are able to predict the next word. Some of the Indian languages like Hindi [3-5], Bengali. For these languages, the next word prediction is implemented. Deep

[1]Professor, Dept.of CSE, R.V.R. & J.C College of Engineering, asrinagesh@gmail.com
[2]Assistant Professor, Dept.of CSE, R.V.R. & J.C College of Engineering, chekuri.sudha@gmail.com
[3]Assistant Professor, Dept.of CSE, R.V.R. & J.C College of Engineering, prasanthiboyapati110@gmail.com
[4]Assistant Professor, Dept.of CSE, R.V.R. & J.C College of Engineering, mails4prk@gmail.com
[5]Assistant Professor, Dept.of CSE, R.V.R. & J.C College of Engineering, prasadsiva_17@yahoo.com

learning [7] is a particular kind of machine learning that achieves power and flexibility by learning to represent the world as a nested hierarchy of concepts.

Deep learning is a sub-branch of machine learning which is completely based on artificial neural networks [6-8], as neural networks are impersonating the human brain, deep learning does not need to write much programming.

II. LITERATURE SURVEY

Natural Language Processing (NLP) is a systematic [2] and significant approach to produce meaningful text that is understandable by humans. For generating the text, the data is collected from different sources or taken as input from the users. There has been a drastic change in the field of NLP over the past few years. Previously, NLP techniques employed shallow machine learning models, which consisted of handcrafted features and were very time-consuming. Due to the increasing popularity of word embedding, neural networks have achieved greater success in comparison to traditional machine learning models.

A. Hassa, et. al., [13] have proposed RNN for the structure, sentence representation. This tree-like structure captures the semantics of the sentences. The text is analyzed word by word by using RNN [9] then the semantics of all the previous texts are preserved in a fixed size hidden layer. For the proposed system LSTM plays an important role, being a memory storage it holds the characters which helps in predicting the next word.

V. Tran, et. al., [10] have proposed that n-gram is a contiguous sequence of ‘n’ items from a given sequence of input text. If the given sentence is considered as ‘S’, we can construct a list of n-grams from ‘S’, by finding pairs of words (POW) that occur next to each other. The model is used to develop the probability of sentences using the chain rule of unconditional probability.

The application of Neural Networks gained popularity in recent times over the traditional methods due to their correctness in generating the text. Neural Networks are inspired by the functioning of the brain and are proving to be efficient models in generating accurate real-world conversational or written context format. RNN[12] was popularly used for text generation because of its ability to process sequential data. But due to its limitation of vanishing gradients, it is being replaced by other neural networks. LSTM, and other versions of LSTM, i.e., Bi-LSTM [9], GRU are being popularly used nowadays for generating text in most of the Indian languages. These models are also being used for other NLP-related tasks like Query auto-completion, story generation.

III. PROPOSED WORK

The process starts with the cleaning of the dataset. Then, the dictionary of unique words is created by splitting the words in the dataset and filtering the unique words which constitute a dictionary. The input file is parsed with the iterator, and unique words are collected. Succeeding, the unique words in the dictionary are mapped to indices. This indicates that the words are not easily processed by neural networks in machine learning, and it is important to map them to the indices, which are easy for neural networks to process.

Set sequence length by dividing the sentence into 3:1 ratio as input x and input y and mapping the first six words in input x. Divide the input file into tensor (dataX) based on the sequence length. Create another tensor (dataY) containing the next words of the sequence, which is in the input file which is the output of the project. The ratio can be of our wish that we can also choose the ratio of 6:1 even if that means six input words and the output is one word which is predicted by the model.

Assign a probability to the content of data using the softmax (activation) function. Select the next word based on the probability of dataY, which will be the predicted word of the sequence. Since the aim of the process is to predict the next word, the predicted word completes the sequence. Each iteration user will be given three options in the output among which the user will have to choose the most appropriate option in the context of its application and the next word prediction will be based on the output chosen.

A. Dataset Description

The dataset was collected from the Twitter API in Telugu Language tweets. The text data can be taken from any book as we are not performing any language translations. The dataset contains text data in the form of sentences. The sentences which have a length of more than three words are taken in the work for further computations.

B. Data Preprocessing

The data preprocessing is about cleaning the text and
Also splitting the text sentences into words and then assigning unique numbers to the words, the process of splitting the sentences into words or into tokens is called tokenization. The tokenization is not about cleaning the text, but gathering more information about the text.

1) **Tokenization:** Tokenization is essentially splitting
A phrase, sentence, paragraph, or an entire text document into smaller units, such as individual words or terms. Each of these smaller units is called tokens. The tokens could be words, numbers, or punctuation marks. In tokenization, smaller units are created by locating word boundaries.

మీరు ఎలా ఉన్నారు

'మీరు' 'ఎలా' 'ఉన్నారు'

Figure 1: Sample Telugu Language Tokens

Before processing a natural language, we need to identify the words that constitute a string of characters. That's why tokenization is the most basic step to proceed with NLP (text data). This is important because the meaning of the text could easily be interpreted by analyzing the words present in the text.

In this paper, we have performed tokenization using the Keras library and this type of tokenization has two methods and they are fit_on_texts, texts_to_sequences which are used to update the internal vocabulary and assign unique numbers to words and then to create a word index dictionary.

C. Layers

Every neural network is constructed from three types of layers and they are the input layer, hidden layers, output layer.

1) **Input Layer:** The input layer used here is
embedding layer which takes the input and passes the input to the hidden layers, the embedding layer will have three parameters to be specified and they are input dimension, output dimension,

input length. The embedding layer used here is the Keras embedding layer that is used for text data to take as input.

2) **Hidden Layer:** A hidden layer in an artificial neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.

3) **Output Layer:** Dense layer is the regular deeply connected neural network layer. It is the most common and frequently used layer. A dense layer does the below operation on the input and returns the output.

D. Activation Functions

An activation function is a function that is added into an artificial neural network in order to help the network learn complex patterns in the data that was considered as a model. When comparing with a neuron-based model that is in our human brains, the activation function is at the end deciding what is to be fired to the next neuron. That is exactly analogous to what an activation function does in an Artificial Neural Network as well. It takes in the output signal from the previous cell and converts it into some form that can be taken as input to the next cell.

1) **Rectified Linear Activation Unit (ReLU):** The rectified linear activation function, or ReLU activation function, is perhaps the most common function used for hidden layers. It is a widely used activation function for the hidden layers.

It is common because it is both simple to implement and effective at overcoming the limitations of other previously popular activation functions, such as Sigmoid and Tanh. It works better than the previously used activation functions. Specifically, it is less susceptible to vanishing gradients that prevent deep models from being trained, although it can suffer from other problems like saturated or “dead” units. The formula is max(o, value).

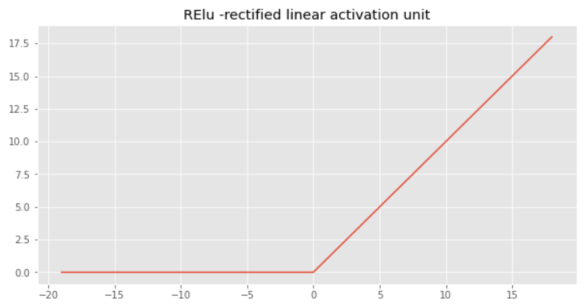

Fig. 2 RELU Curve

2) **Softmax Output Function:** The softmax function Outputs a vector of values that sum to 1.0 that can be interpreted as probabilities of class membership. It is related to the argmax function that outputs a 0 for all options and 1 for the chosen option. Softmax is a “softer” version of argmax that allows a probability-like output of a winner-take-all function.

As such, the input to the function is a vector of real values and the output is a vector of the same length with values that sum to 1.0 like probabilities.

The softmax function is calculated as follows: e^x / sum(e^x). Where x is a vector of outputs and e is a mathematical constant that is the base of the natural logarithm.

E. Model Building

In model building, the model that we have chosen to build the task is LSTM which is an RNN.

1) Long Short Term Memory (LSTM): Long-Term Short Term Memory (LSTMs) is a variant of RNNs which are capable of learning Long Term Dependencies, and thus are widely used for Natural Language processing. LSTMs have memory and remember past data from the input for longer durations of time. They can selectively remember or forget things. They are well suitable for writing data inputs, as any word in a sentence is related to the words around it (previous and upcoming words). Each repeating module in LSTM consists of 4 neural network layers that interact with each other. The cell state in LSTMs decides when to read, write and what should be stored in memory. It has gates that conditionally let the information pass through them, adding or removing it from the cell. Gates works using point-wise multiplication, point-wise addition, and sigmoid function. The output is from 0 to 1 signifying how much information should pass through, where 0 represents no information passing through, and 1 means all information passes through LSTM.

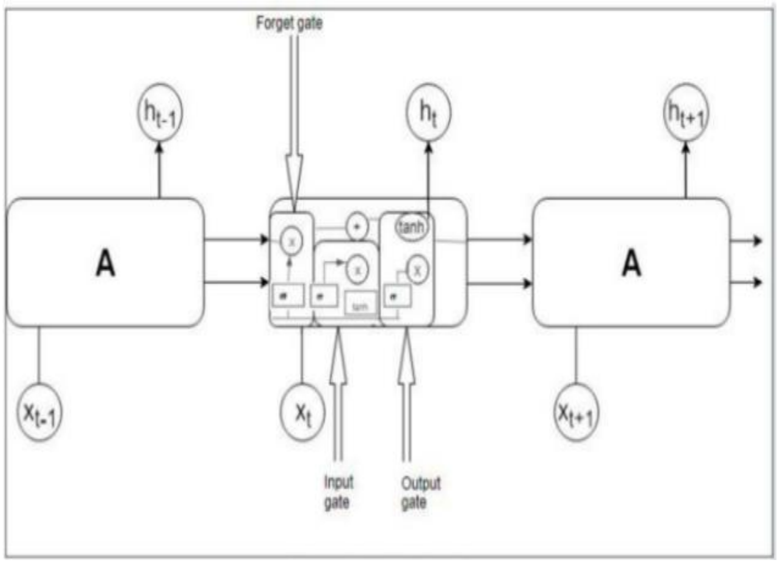

Fig. 4: Proposed LSTM Architecture

If we observe the above figure each cell is having two arrows to the next cell one is called the cell state and the other is called the hidden state. Cell state is also known as long term memory and hidden state is known as short term memory. Where xt-1 is the input at each cell at a particular time-stamp ‘t’. The entire process flow diagram of this project is represented in figure 4.

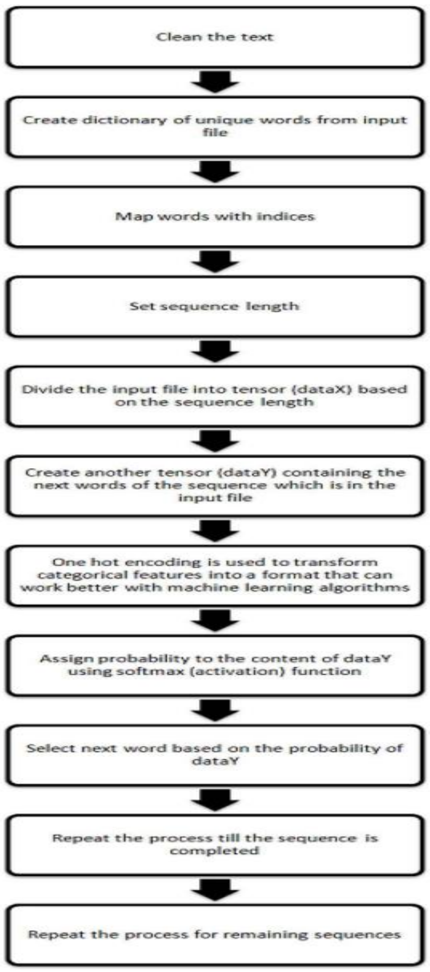
Clean the text
Create dictionary of unique words from input file
Map words with indices
Set sequence length
Divide the input file into tensor (dataX) based on the sequence length
Create another tensor (dataY) containing the next words of the sequence which is in the input file
One hot encoding is used to transform categorical features into a format that can work better with machine learning algorithms
Assign probability to the content of dataY using softmax (activation) function
Select next word based on the probability of dataY
Repeat the process till the sequence is completed
Repeat the process for remaining sequences

Fig. 4 Process flow diagram of the Proposed Approach

IV. RESULT ANALYSIS

The collected corpus from Twitter is 11,456 tweets. English-Telugu code mixed tweets count is 6,234. Manually sentiment labelling is assigned as positive, negative and neutral. The Required pre-processing steps are applied to the collected code mixed data. All punctuation symbols, duplicate tweets, RT, tags, extra symbols, and emphasize words are in code mixed data. English and Telugu stop word list and all unnecessary text is eliminated. Some of the text is replaced with suitable text. In the sample dataset, with cleaning and pre-processing of raw data. The Twitter data which are collected randomly from the Twitter API from November 2020 to February 2021 is related to Andhra Pradesh politics

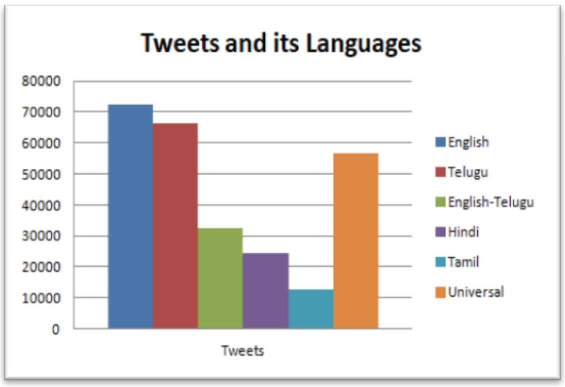

Figure 5: Multi-lingual Tweets and languages.

Figure 5 explains the collected tweets and their languages, English-Telugu tweets along with the different language tweets.

The collected corpus from Twitter is 16,124 tweets. English-Telugu code mixed tweets count is 2,345. Manually sentiment labelling is assigned as positive, negative and neutral. The Required pre-processing steps are applied to the collected code mixed data. All punctuation symbols, duplicate tweets, RT, tags, extra symbols, and emphasize words are in code mixed data. English and Telugu stop word list and all unnecessary text is eliminated. Some of the text is replaced with suitable text. Figure 4 is a sample of the dataset, with cleaning and pre-processing of raw data.

In Result Analysis the accuracy of the model obtained is shown in the diagram. The trained and validation accuracy of the model achieved is shown in the below figure.

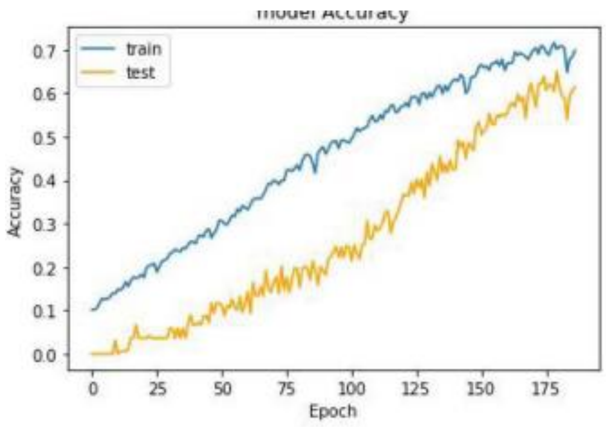

Fig. 6: Accuracy of the Proposed LSTM method

Here we can observe that the accuracy of LSTM is 70% and we can also observe that as the number of epochs increases the accuracy also increases. The number of epochs is not fixed so that we can change the value of epochs as we need.

The author that I took reference for this project got accuracy 68% by using LSTM. We actually got 70% by using LSTM.

TABLE I

Comparison of Accuracy

omparison ccuracy	**STM**
ccuracy	3%
roposed Method ccuracy	0%

V. CONCLUSION

A Machine learning model to predict the next word in a given sequence of words was built using LSTM models. It was then extended to predict the next few words in the same sequence. The model developed can be used for predicting the next word in the Telugu Language. This can effectively reduce the number of words that the user has to type, thus increasing the typing speed of the text inputs. It also helps in minimizing the spelling mistakes that are inputted by the user.

VI. FUTURE SCOPE

In the future, the system can be extended for Telugu natural language generation tasks like story auto-completion, poem auto-completion, etc. The model is limited to a specific dataset and more randomness can be incorporated into the model by enhancing the scope.

The system can be adapted to new words that are not part of its vocabulary. This adaptation will be done when the model encounters a new word and adding the word to the vocabulary. This way the model becomes more generalized. By remembering the history of the user's model can become personalized. The system can be personalized to predict words based on the user's history.

REFERENCES

[1] P. P. Barman and A. Boruah, "A run-based approach for next word prediction in Assamese

[2] Phonetic transcription," Sth International Conference on Advances in Computing and Communication, 2018.

[3] C. McCormick, Latent Semantic Analysis (LSA) for Text Classification Tutorial, 20 19 (accessed February 3, 20 19). http://mccormickml.comJ 20 16/03/25/1sa-for-text-classification-tutorial/.

[4] N. N. Shah, N. Bhatt, and A. Ganatra, "A unique word prediction system for text entry in Hindi," in Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies, p. 118, ACM, 20 16.

[5] M. K. Sharma and D. Smanta, "Word prediction system for text entry in Hindi," ACM Trans. Asian Lang. Inform. Process, 06 2014.

[6] R. Devi and M. Dua, "Performance evaluation of different similarity functions and classification methods using web-based Hindi language question answering system," Procedia Computer Science, vol. 92, pp. 520-525, 20 16.
[7] S. Lai, L. Xu, K. Liu and J. Zhao, “Recurrent Convolutional Neural Networks for Text Classification”, Proceedings of the Twenty-Ninth AAAI Conference on AI 2015.
[8] P. Ongsulee, “Artificial Intelligence, Machine Learning and Deep Learning”, 15th International Conference on ICT and Knowledge Engineering (ICT&KE), 2017.
[9] W. Yin, K. Kann, Mo Yu and H. Schütze, “Comparative study of CNN and RNN for Natural Language Processing”, Feb-17.
[10] Z.Shi, M. Shi and C. Li, “The prediction of character based on Recurrent Neural network language model”, IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017.
[11] V. Tran, K. Nguyen and D. Bui, “A Vietnamese Language Model Based on Recurrent Neural Network”, 2016 Eighth International Conference on Knowledge and Systems Engineering.
[12] J. Lee and F. Dernoncourt, “Sequential Short-Text Classification with Recurrent and Convolutional Neural Networks”, Conference paper at NAACL 2016.
[13] M. Liang and X. Hu, “Recurrent Convolutional Neural Network for Object Recognition”, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015.
[14] A. Hassan and A.Mahmood, “Deep Learning for Sentence Classification”, IEEE Long Island Systems, Applications and Technology Conference (LISAT), 2017.