Report

On

Next Word Prediction

Prepared by

Bhargav Bechara (18012011004)

Guided By

Prof. Ketan J. Sarvakar

B.Tech Semester VII

Nov-Dec -2021

Subject: Deep Learning

Subject code: 2CEIT78PE1

Department of Computer Engineering

U. V. Patel College of Engineering

Ganpat University, Ganpat Vidyanagar – 384012

Abstract:

Deep learning has been an area of research and used widely in different applications. We often love texting each other and find that whenever we try to type a text a suggestion poops up trying to predict the next word we want to write. This process of prediction is one of the applications NLP and DL deals with. We have made huge progress here and we can use Recurrent neural networks for such a process. As we type in what is the weather we already receive some predictions. We can see that certain next words are predicted for the weather. The next word prediction for a particular user's texting or typing can be awesome. It would save a lot of time by understanding the user's patterns of texting. This could be also used by our virtual assistant to complete certain sentences. Overall, the predictive search system and next word prediction is a very fun concept.

Problem Description:

Most of the keyboards in smartphones give next word prediction features; google also uses next word prediction based on our browsing history. So a preloaded data is also stored in the keyboard function of our smartphones to predict the next word correctly. In this project, I will train a Deep Learning model for next word prediction using Python. I will use the Tensor flow and Keras library in Python for next word prediction model. For making a Next Word Prediction model, I will train a Recurrent Neural Network (RNN).

Objectives:

- Exploratory analysis - perform a thorough exploratory analysis of the data, understanding the distribution of words and relationship between the words in the corpora.
- Understand frequencies of words and word pairs - build figures and tables to understand variation in the frequencies of words and word pairs in the data.
- Build basic n-gram model - using the exploratory analysis you performed, build a basic n-gram model for predicting the next word based on the previous 1, 2, or 3 words.
- Build a model to handle unseen n-grams - in some cases people will want to type a combination of words that does not appear in the corpora. Build a model to handle cases where a particular n-gram isn't observed.

Notebook used:

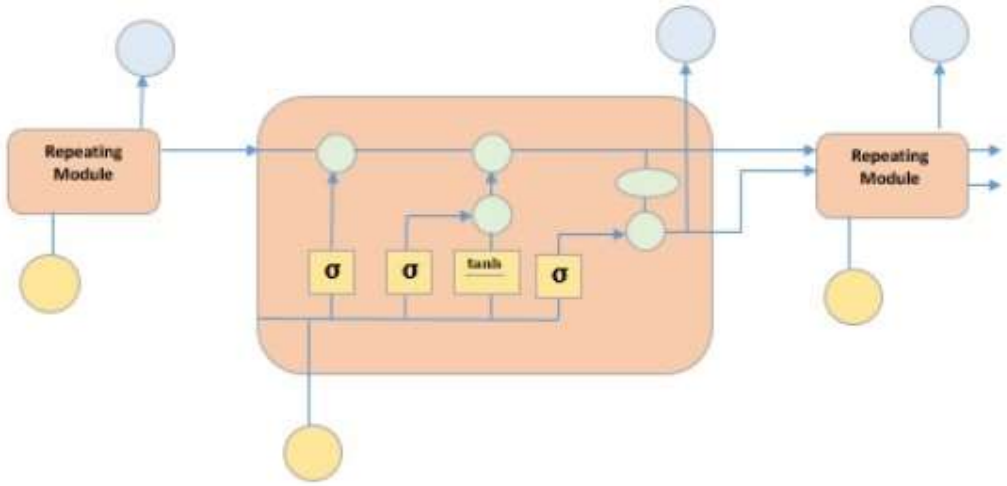
Google Colab - Colaboratory, or "Colab" for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education.

Technical requirements:

Sr no.	Name	Description
1	RNN	A recurrent neural network (RNN) is a type of artificial neural network which uses sequential data or time series data. These deep learning algorithms are commonly used for ordinal or temporal problems, such as language translation, natural language processing (nlp), speech recognition, and image captioning.
2	LSTM	Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feedforward neural networks, LSTM has feedback connections. It can process not only single data points (such as images), but also entire sequences of data (such as speech or video).
3	TensorFlow	TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.
4	Keras	Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library.

Architecture of LSTM:

It is special kind of recurrent neural network that is capable of learning long term dependencies in data. This is achieved because the recurring module of the model has a combination of four layers interacting with each other.



The picture above depicts four neural network layers in yellow boxes, point wise operators in green circles, input in yellow circles and cell state in blue circles.

An LSTM module has a cell state and three gates which provides them with the power to selectively learn, unlearn or retain information from each of the units.

The cell state in LSTM helps the information to flow through the units without being altered by allowing only a few linear interactions. Each unit has an input, output and a forget gate which can add or remove the information to the cell state.

The forget gate decides which information from the previous cell state should be forgotten for which it uses a sigmoid function.

The input gate controls the information flow to the current cell state using a point-wise multiplication operation of 'sigmoid' and 'tanh' respectively. Finally, the output gate decides which information should be passed on to the next hidden state

Work Flow:

Step 1: Import Required libraries
Step 2: Read the corpus from txt file
Step 3: Pre-process the corpus
Step 4: Tokenizing the text into words
Step 5: Set length of the sequence to train
Step 6: Converting the texts into integer sequence
Step 7: Build lstm model
Step 8: Train the model
Step 9: Prediction

Dataset Name:

nextword.txt

Dataset location:

https://drive.google.com/drive/folders/1V1dKCZQoNcV1x8LsBpVH0yQMZ2nZNhNt?usp=sharing

GitHub Code link:

https://github.com/Bhargav9900/Next-Word-Prediction_DL

Output:

```
input_text="next word"
print("Possible next word will be:")
predict_next_word(input_text)
```

```
Possible next word will be:
['prediction']
```

```
input_text="Language Modeling"
print("Possible next word will be:")
predict_next_word(input_text)
```

```
Possible next word will be:
['and']
```