**Project Name:** Next Word Prediction Model

**Github Link:** https://github.com/utkrisht2000/utkrisht2000-Next-Word-Prediction-Model.git

**Why was this project created?**

Long phrases might be tedious to write, but text prediction technology built into keyboards makes this simple. Another name for next word prediction is language modeling. It's the endeavor of predicting what word comes straightaway. It has numerous applications and is one of the main tasks of human language technology.

**What problem is it solving?**

Our goal is to make predictions about the following words in sentences based on the words that came before and a corpus used to train the model. In order to anticipate the following word in a sequence, we have trained neural networks and n-gram language models.

**Entire explanation of project**

* **PROPOSED APPROACH**

A variety of inputs are obtained, and the following stage is to purge the dataset of any extraneous information. The dataset's beginning and finish will be removed. This information is unimportant to us. We then proceed to replace all extra new lines that are superfluous, the carriage return, and the Unicode character. By converting each text in a corpus into either a series of integers or a vector with a coefficient for each token that may be binary based on word count or based on tf-idf, the Keras Tokenizer enables us to vectorize a corpus of text. We'll create a sequential model. Next, we will set the input and output dimensions for an embedding layer. Given that the prediction will be based on just one word and that word will be the subject of the response, it is crucial to set the input length as 1. After that, we will extend our design with an LSTM layer. We'll give it 1000 units and be careful to mark the sequences as true when we return them. We will also transmit it through an additional 1000 units for the subsequent LSTM layer, but we don't need to provide the return sequence because it is false by default. With relu set as the activation, we will use the dense layer function to send this via a hidden layer with 1000 node units. Finally, we run it through an output layer with a softmax activation and the chosen vocabulary size. We are given a large number of possibilities for the outputs that are equivalent to the vocabulary size thanks to the softmax activation.

Algorithm for creating next word prediction model :

**Step 1:** Dataset is imported

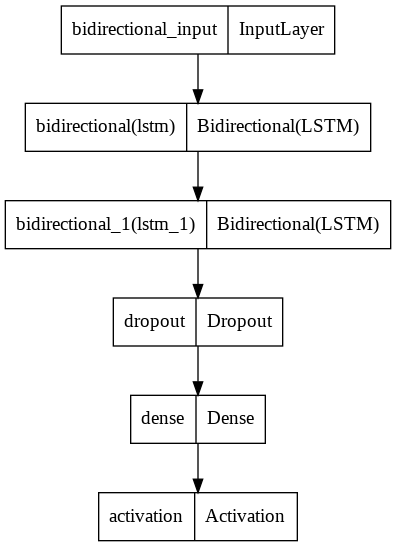
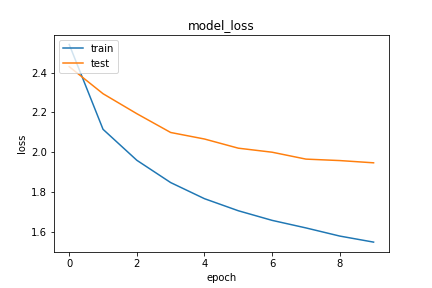
**Step 2:** The data is preprocessed, and the dataset is split into training and testing.

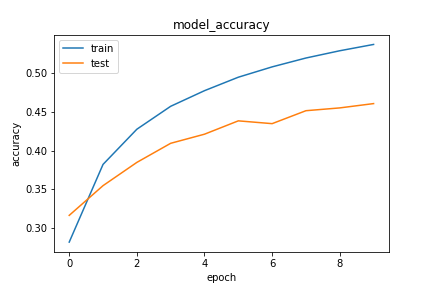
**Step 3:** Text Analysis

**Step 5:** Tokenization

**Step 6:** Pad Sequences

**Step 7:** Compile and Fit

* **DATA FLOW DIAGRAM**
* **RESULT**



* **CONCLUSION**

On the basis of the present dataset, this word prediction model is fairly accurate. Applying multiple pattern-discovery techniques is necessary for NLP in order to get rid of noisy data. In roughly one hundred epochs, the loss was greatly decreased. The processing of huge files or datasets still requires considerable optimization. To improve the model's prediction, however, bound preprocessing processes and bound model adjustments are frequently developed.