**ABSTRACT :**

Text generation is an important direction in the field of natural language processing (NLP). In the era of pre-training model, transformer improved pre-training text generation model still can not achieve relatively ideal results, and at the same time, there is no efficient language model to automatically evaluate the quality of generated text. As an improved variant of cyclic neural network (RNN), long-term memory network (LSTM) is characterized by its long-term dependence, and it performs very well in the task of processing long **sequences**. The LSTM has many improved variants for different tasks, including the gated loop unit (GRU) and the LSTM with a peephole connection, all of which have better performance than the LSTM in specific tasks. However, it is not clear whether these improved variants have better performance in the field of text generation. Therefore, an exploratory text generation experiment is conducted to solve this problem. By comparing the generated text quality of standard LSTM with LSTM's improved variant GRU and LSTM model with visual hole connection, the evaluation results of LSTM model in long text field are obviously better than those of the other two models through three evaluation indexes: confusion degree, BERT score and BLEURT. Finally, we draw a conclusion and research direction that the native LSTM in the field of long text still has very superior performance. In the future, we can design a pre-training model based on LSTM for text generation. Future language models can be designed to guide the optimization and improvement of language models through large-scale evaluation using automated evaluation indicators such as BERT score and BLEURT, which are close to manual evaluation, so as to design language models that can generate higher quality text.

**Keywords :**

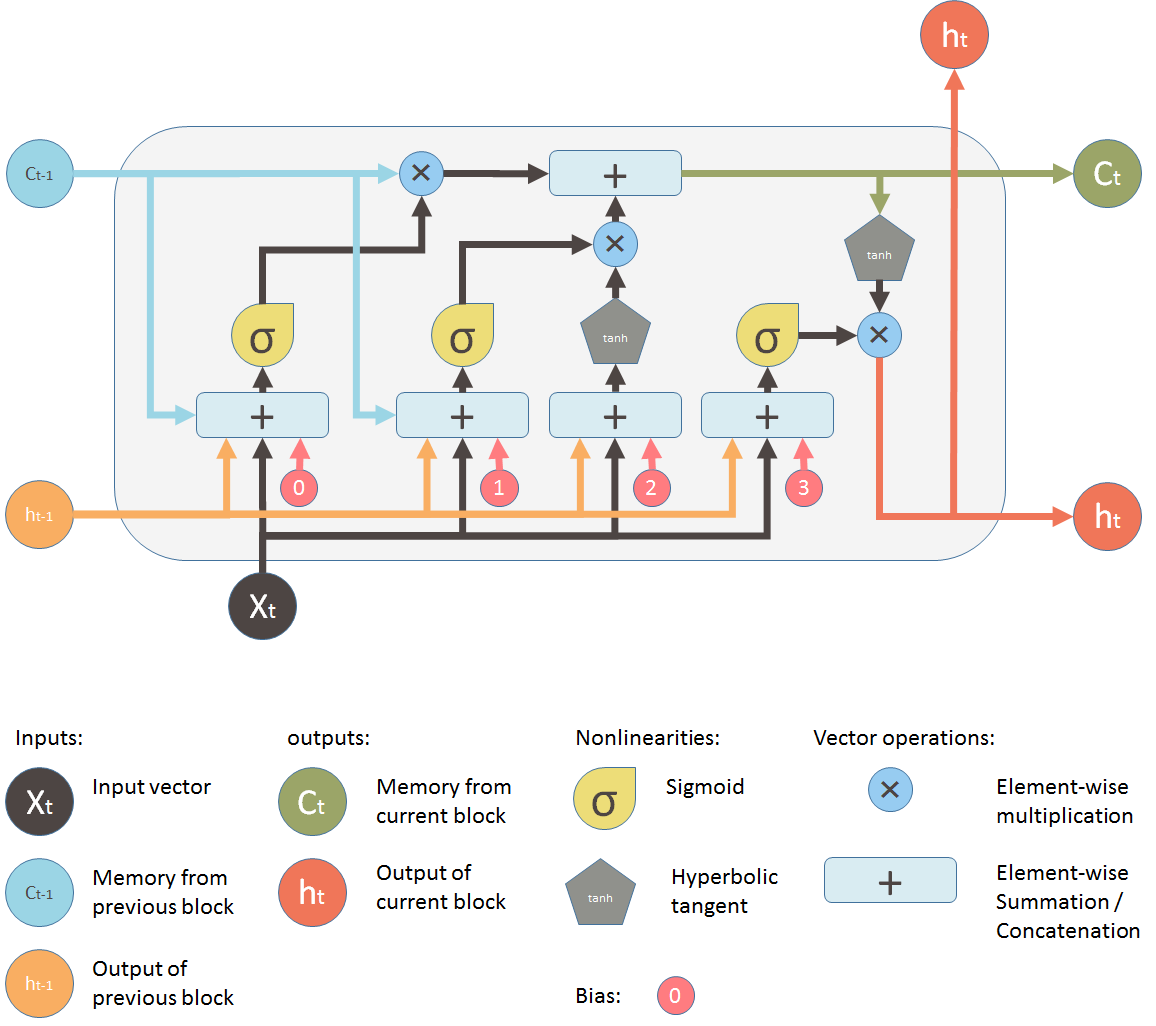
LSTM; Peephole Connection; GRU; BERT Score; BLUERT.

**1 . INTRODUCTION**

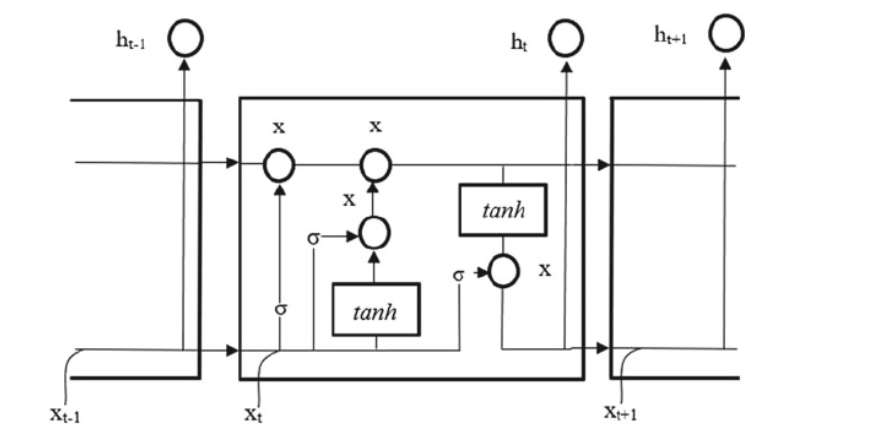
Text generation is an important aspect of natural language processing, and it is the most difficult and difficult scientific problem to explain. In recent years, the application and demand of text generation has been increasing. Tencent's Dream writer script-writing robot and today's headline-making xiaoming bot news-writing robot and other text-generation robots are constantly emerging. For a moment, they have become one of the hottest topics. The principle of machine news writing is to use the neural network model to generate new news stories based on old news texts and data [1] . Automatic text generation can include text-to-text generation (text-to- text generation), meaning-totext generation (text generation), data-to-text generation (data-to- text generation) and image-to-text generation (image-to- text generation)[2].Among them, the principle of text-to-text generation technology is to learn the existing old text to generate new text, such as to transform the characters and plots of the old story set into a more vivid and interesting new story, to realize the processing and utilization of text. This text generation experiment uses text-to-text generation.

Although the field of text generation is developing rapidly and various model structures emerge endlessly, the quality of generated text is still difficult to reach the level of human writing and there is no uniform evaluation standard. At present, the model with the highest quality of text generation is the pre-training model GPT-3 with transformer decoding structure. Although the GPT series model has achieved good results, when putting the text generated by GPT and the normal text written manually together, one can easily determine which author the text belongs to, which indicates that the current language model can not pass the Turing test [3].But for the generation of prose, novels and other long texts containing human emotions, the current research is still in the primary stage. Deep and high-parameter GPT model fails to solve these problems, which also makes the development of text generation almost stagnate. There are two main reasons: one is that GPT based on transformer is inefficient to deal with the strict sequence of tasks before and after the text, and there is structural difference between the base model and the specific field. The second is the lack of effective evaluation indicators to guide the optimization and improvement of the model. In many cases, researchers are unable to determine whether their improvement measures really lead to the improvement of the model performance. When the number of texts is large, manual evaluation is faced with a series of problems caused by excessive time cost. In fact, the difficulty of evaluating language models has always been the key factor restricting the development of natural language generation (NLG), and the accuracy and flexibility of manual evaluation has always been the most convincing evaluation index, but manual evaluation has a great defect of slow evaluation speed. As the most commonly used evaluation index of language model, the degree of confusion is often used as an important reference because the evaluation is relatively objective and fast, but sometimes the text with low degree of confusion but poor quality is evaluated manually. When the number of text is large, manual evaluation will consume a lot of time, which makes the model training have to stop halfway, which directly leads to the development of language model subject to the slow speed of manual evaluation. In order to make the automatic evaluation result of the machine closer to the manual evaluation result, there are new evaluation indexes based on the pre-training model BERT, such as ERT score, BLEURT, and the accuracy of these new evaluation indexes has been proved in many experiments. The earliest improvements in text generation are actually the use of the improved version of LSTM of RNN, which has unique three-door and two-state structural features that make it stand out in the NLP field. In the development history of LSTM, there are many improved versions for LSTM, such as GRU, LSTM with peephole, which perform better in some specific areas. In addition, the poor parallelism of LSTM makes it cold in the training model era. As technology continues to evolve and new changes have emerged in many studies, we need to revisit the role of LSTM. This study creatively evaluated the quality of text generated by different models using new evaluation metrics, and aimed to find the best base model by comparing the performance of LSTM with its variant GRU and LSTM with visual hole connections in the long text field. In order to ensure that the generated text can be compared at a high quality level and to maximize the subtle performance differences between different models, CBOW is also used in all models to improve the quality of word vectors, and cluster search is used to aid prediction.

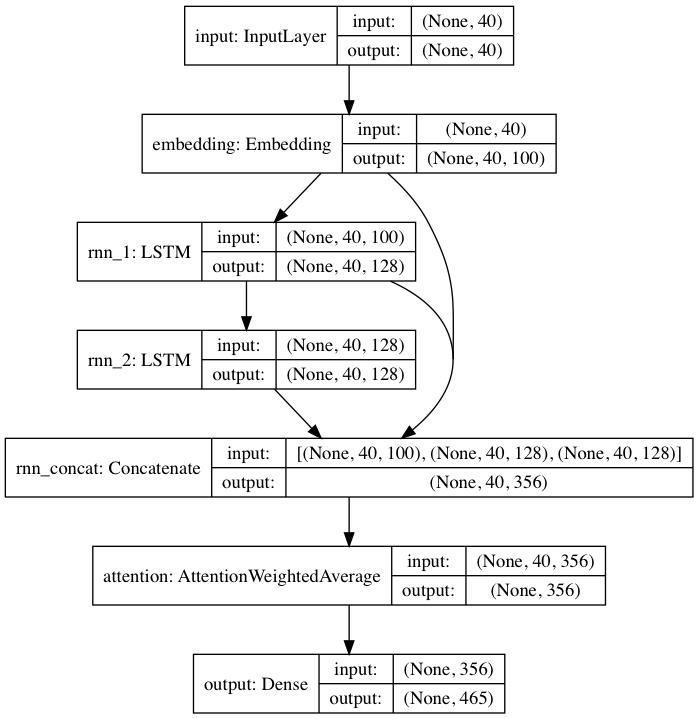
**2 .**  **THEORITICAL ANALYSIS :**



**Fig. :**   **OUTPUT GATE**



**Fig. : Working of a long short-term memory network**



**Fig. : Scientific Diagram of LSTM Model for Text Generation**

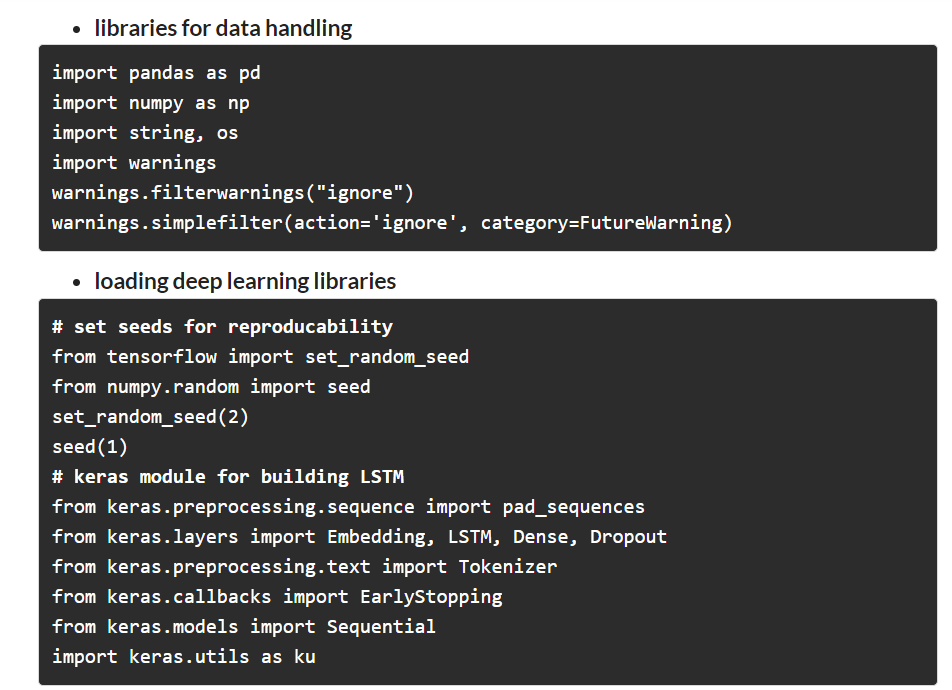
**3 . EXPERIMENTAL INVESTIGATIONS :**

## Implementing Text Generation

There are steps various steps listed for text generation:-

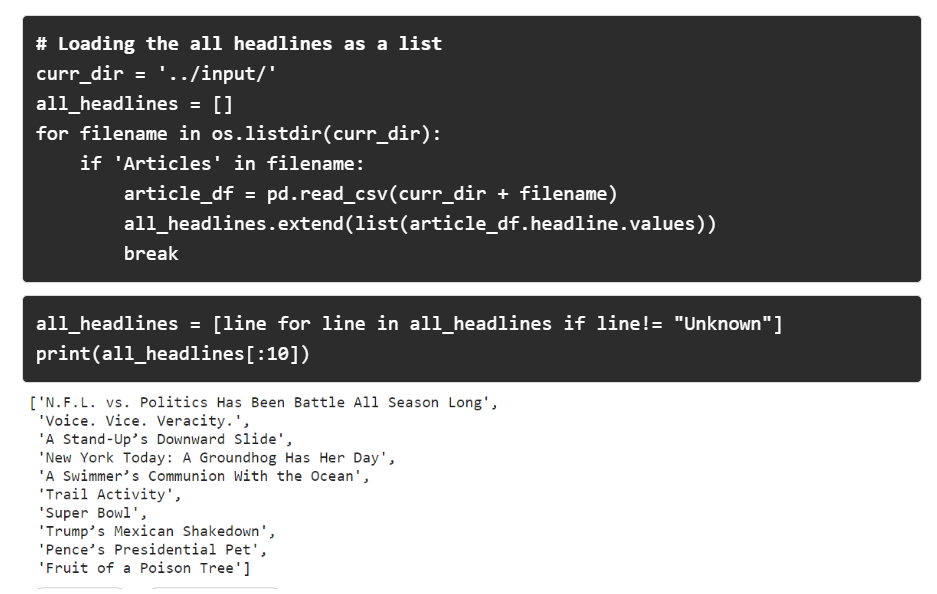
* Load the necessary libraries
* Load the textual- data
* Perform text-cleaning if needed
* Data preparation for training
* Define and train the LSTM model
* Prediction

#### **Loading necessary libraries :**



#### **Loading the Dataset :**

The dataset contains various articles and comments. Our objective is to load all the articles as headlines and merge them into a list.

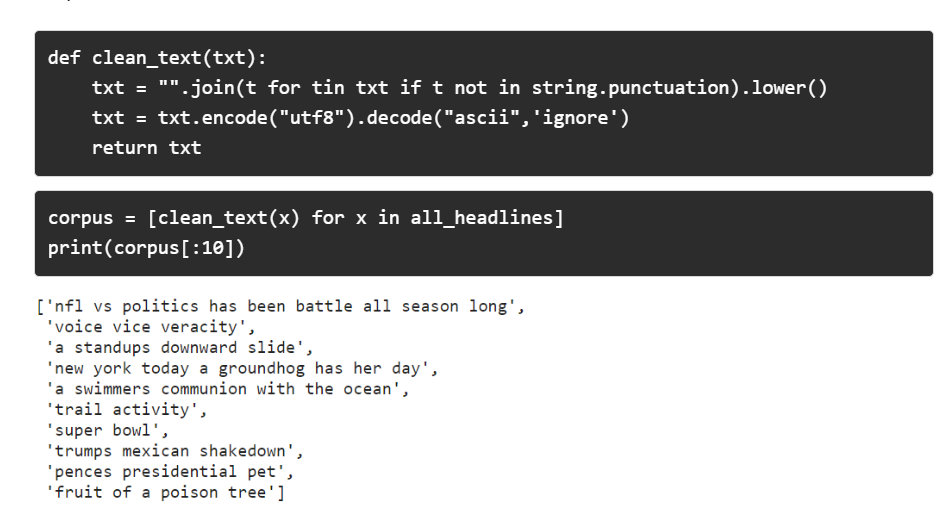


#### **Dataset Preparation :**

For Dataset Preparation our first task will be to clean the text data which includes removing punctuations, lowercasing words, etc.

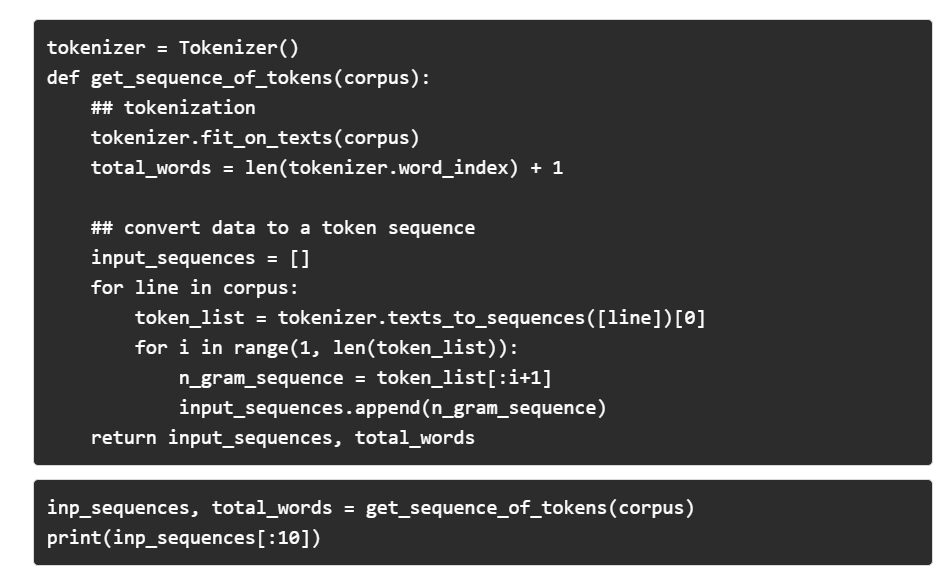
* **Data Cleaning :**

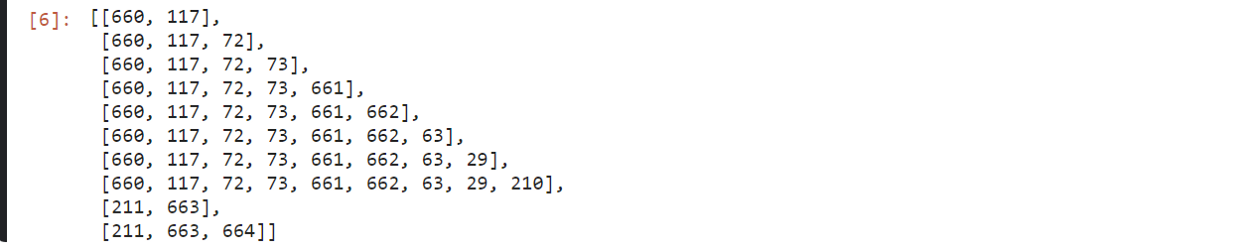
We defined a function that takes a single headline at a time and returns the cleaned headline. Using iteration we have passed each headline and made a list of cleaned data corpus.



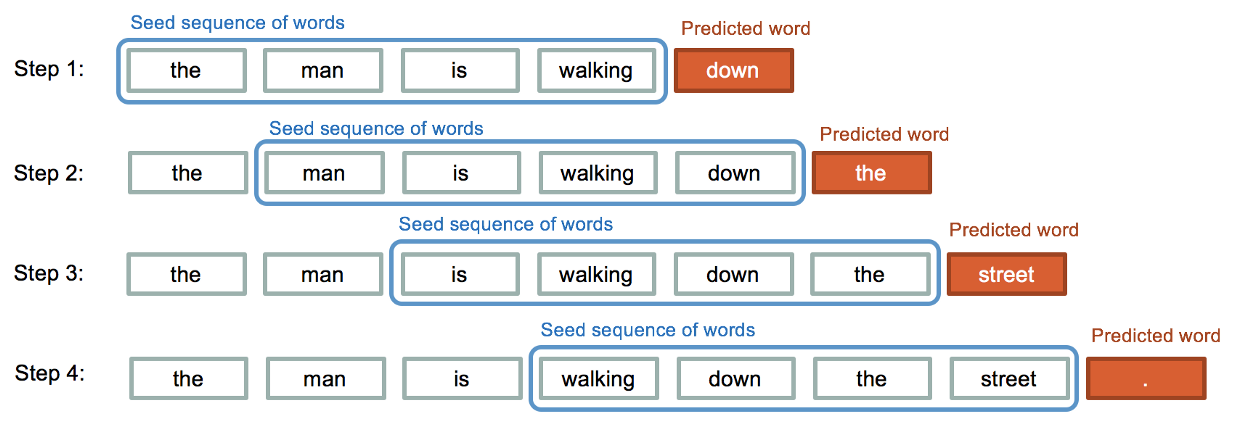
* **Generating n-gram Sequence for training :**

In NLP language model requires sequential input data, and input word/token must be numerical. Here we are generating n-grams in order to train our model for next word prediction .





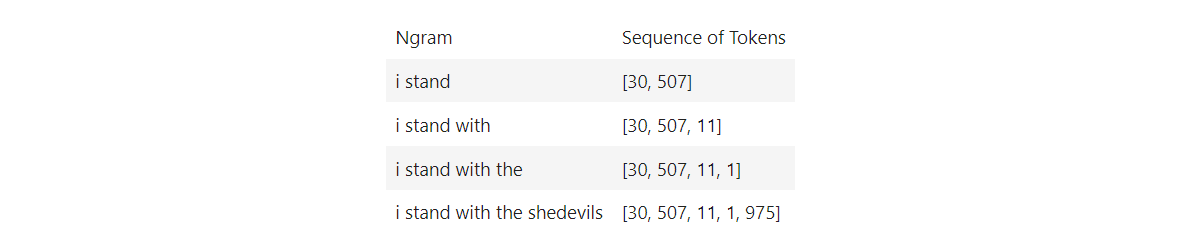
As you see that **inp\_sequence** is an n-gram sequence that is required for training next-word prediction. we had 829 headlines and using the n-gram concept we have now 4544 rows.

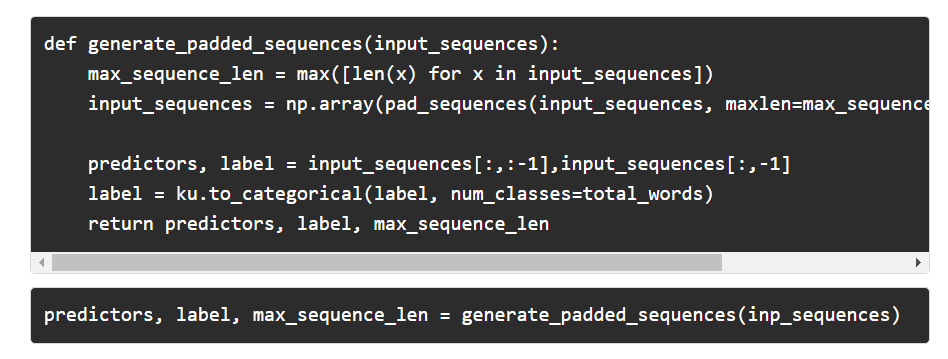


You can relate the **inp\_sequences** with this picture where you can clearly see that in every step we add a token to the Seed sequence for training.

* **Padding the Sequences**

The **inp\_sequence** we just made have variable sequence length, which is not favorable for training, using padding we make every sequence of having the same length.





* **predictors** : these are tokens that will be used as input for predicting the next word.
* **label:** is the next word to be predicted.
* **max\_sequence\_len:** is the sequence length.
* **pad\_sequence:** provided by Keras is used to pad an array of tokens to a given length.
* In this case,**max\_sequence\_len** is 17.

## LSTMs for Text Generation :

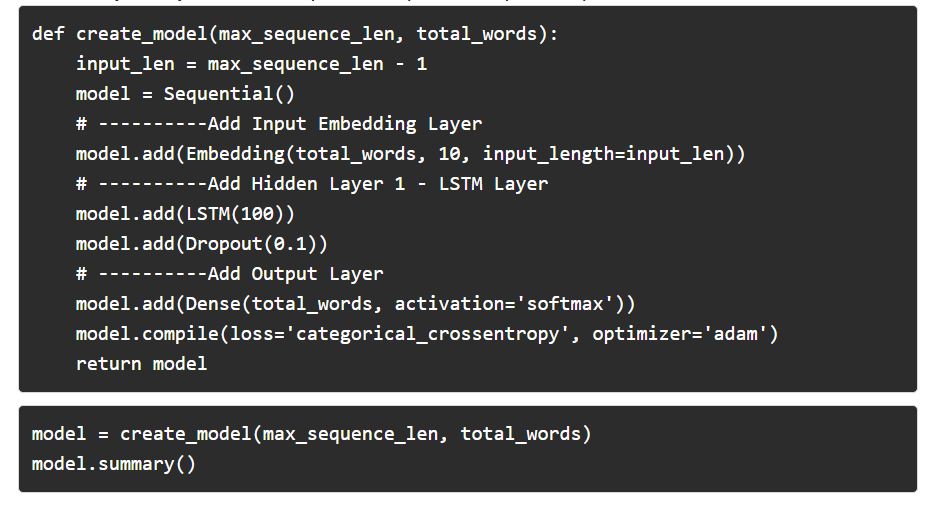
#### Model Creation

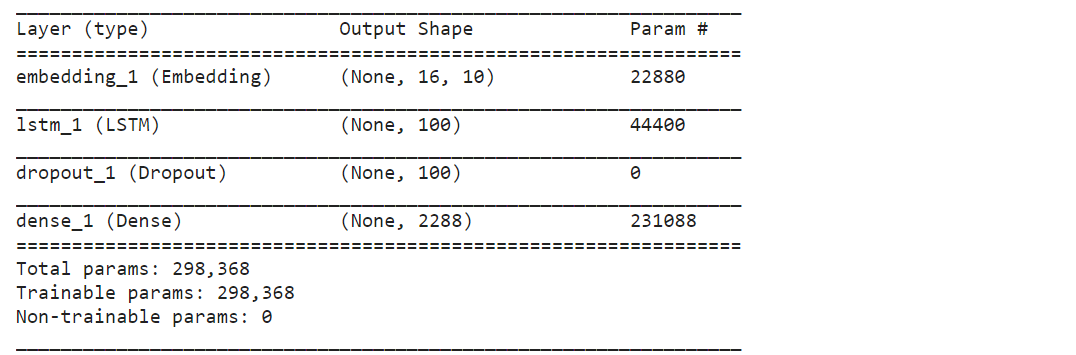
So far we have prepared the data for training. now in this step, we will create an LSTM model that will take **predictors** as input X and **labels** as input y.

**A quick reminder on Layers in Keras:-**

* **Input Layer**: This is responsible for taking input sequence.
* **LSTM Layer**: It calculates the output using LSTM units and returns hidden and cell states. In our case we have added 100 units in the layer, that can be fine-tuned later.
* **Dropout Layer**: This layer is responsible for regularisation which means it prevents over-fitting. this is done by turning off the activations of some neurons in the LSTM layer.
* **Output Layer**: This Computes the probability of our prediction.

We will run this model for total 100 epoochs but it can be experimented further.





#### Training the model :

After building the model architecture we can train the model using our **predictors** (X\_train) and **label**(y\_train).100 epochs should be enough.



Epoch 1/100  
Epoch 2/100  
Epoch 3/100  
Epoch 4/100  
Epoch 5/100  
Epoch 6/100  
Epoch 7/100  
Epoch 8/100  
Epoch 9/100  
Epoch 10/100  
Epoch 11/100  
Epoch 12/100  
Epoch 13/100  
Epoch 14/100  
Epoch 15/100  
Epoch 16/100  
Epoch 17/100  
Epoch 18/100  
Epoch 19/100  
Epoch 20/100  
Epoch 21/100  
Epoch 22/100  
Epoch 23/100  
Epoch 24/100  
Epoch 25/100  
Epoch 26/100  
Epoch 27/100  
Epoch 28/100  
Epoch 29/100  
Epoch 30/100  
Epoch 31/100  
Epoch 32/100  
Epoch 33/100  
Epoch 34/100  
Epoch 35/100  
Epoch 36/100  
Epoch 37/100  
Epoch 38/100  
Epoch 39/100  
Epoch 40/100  
Epoch 41/100  
Epoch 42/100  
Epoch 43/100  
Epoch 44/100  
Epoch 45/100  
Epoch 46/100  
Epoch 47/100  
Epoch 48/100  
Epoch 49/100  
Epoch 50/100  
Epoch 51/100  
Epoch 52/100  
Epoch 53/100  
Epoch 54/100  
Epoch 55/100  
Epoch 56/100  
Epoch 57/100  
Epoch 58/100  
Epoch 59/100  
Epoch 60/100  
Epoch 61/100  
Epoch 62/100  
Epoch 63/100  
Epoch 64/100  
Epoch 65/100  
Epoch 66/100  
Epoch 67/100  
Epoch 68/100  
Epoch 69/100  
Epoch 70/100  
Epoch 71/100  
Epoch 72/100  
Epoch 73/100  
Epoch 74/100  
Epoch 75/100  
Epoch 76/100  
Epoch 77/100  
Epoch 78/100  
Epoch 79/100  
Epoch 80/100  
Epoch 81/100  
Epoch 82/100  
Epoch 83/100  
Epoch 84/100  
Epoch 85/100  
Epoch 86/100  
Epoch 87/100

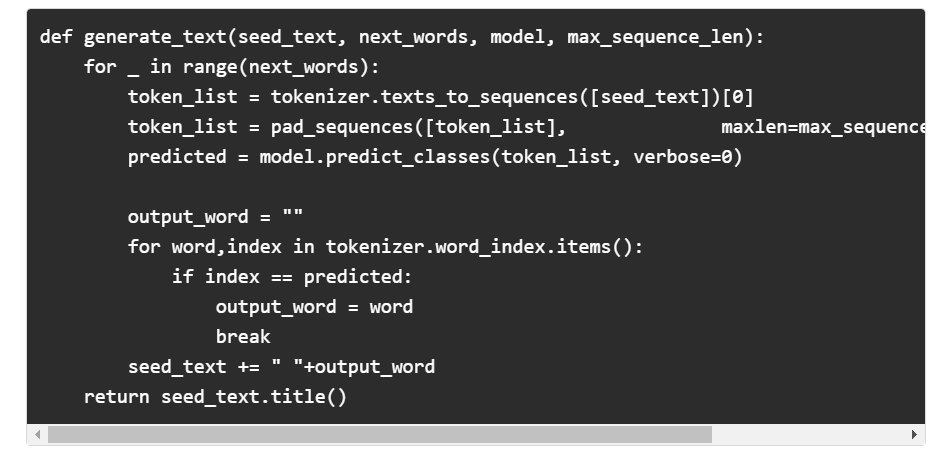
Epoch 88/100  
Epoch 89/100  
Epoch 90/100  
Epoch 91/100  
Epoch 92/100  
Epoch 93/100  
Epoch 94/100  
Epoch 95/100  
Epoch 96/100  
Epoch 97/100  
Epoch 98/100  
Epoch 99/100  
Epoch 100/100

Out[7]:

<keras.callbacks.History at 0x7f2ddf540550>

#### Text Generation (Prediction) :

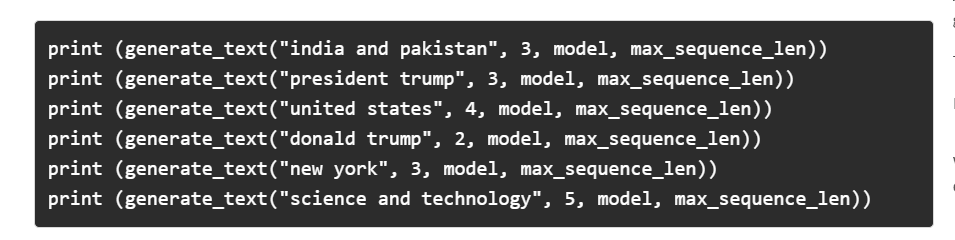
We have trained our model architecture and now it’s ready to generate text. We need to write a function to predict the next word based on the input words. We also have to tokenize the sequence and pad it with the same **sequence\_length** we provided for training, and then we will append each predicted word as a string.



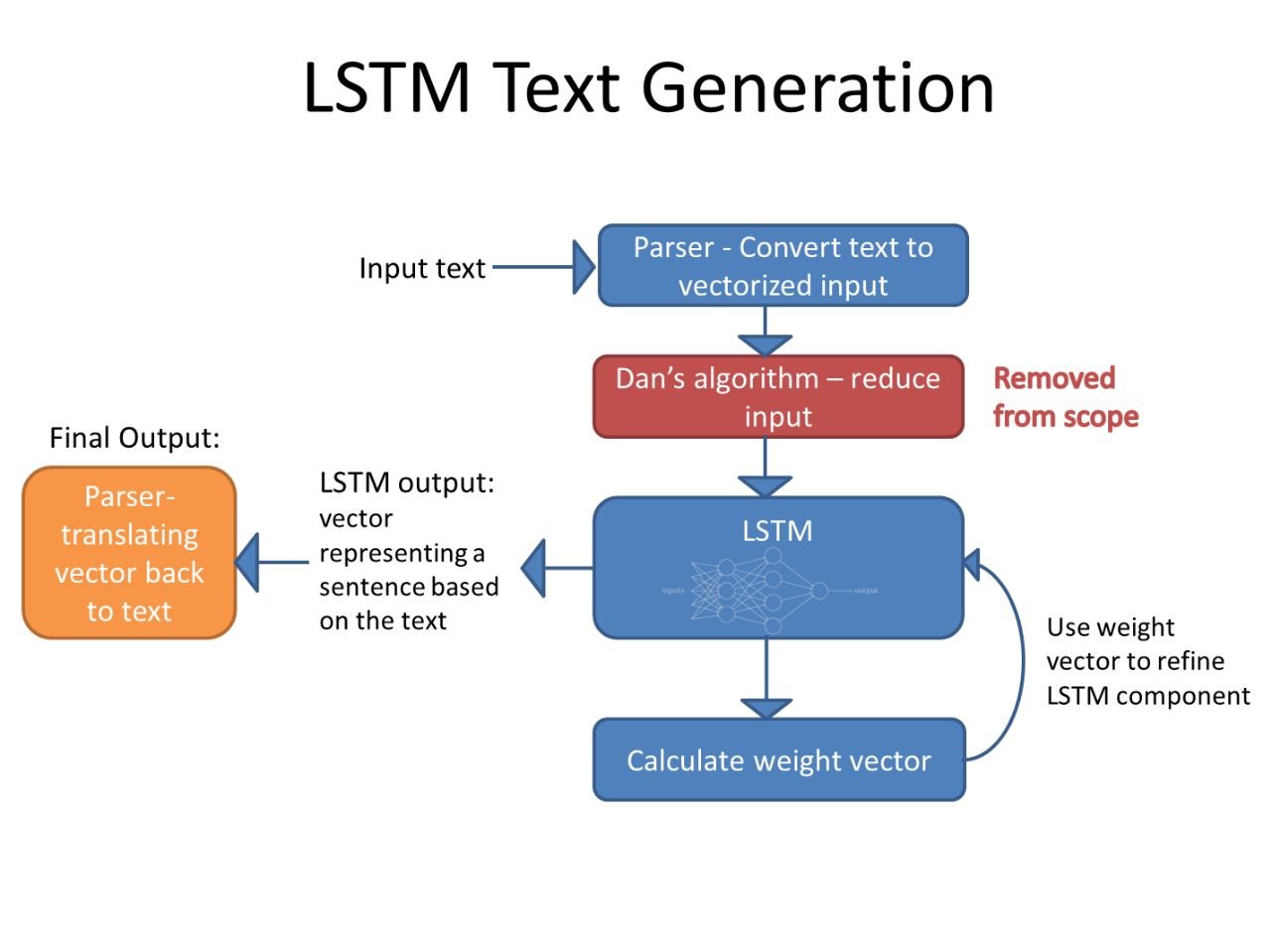
* **seed\_text** : it’s the initial words that will be passed for text generation.
* **predict\_classes:** it will return the token id for the predicted word.
* **predicted**: Its token id for predicted word and this will be converted back into a word using the dictionary**tokenizer.word\_index .items()**.
* **next\_words** It’s the number of next words we want to be predicted.

#### **Prediction :**

Calling the function **generate\_text**will generate text.generate\_text function takes initial words and number of words to be predicted, model name, and sequence length.

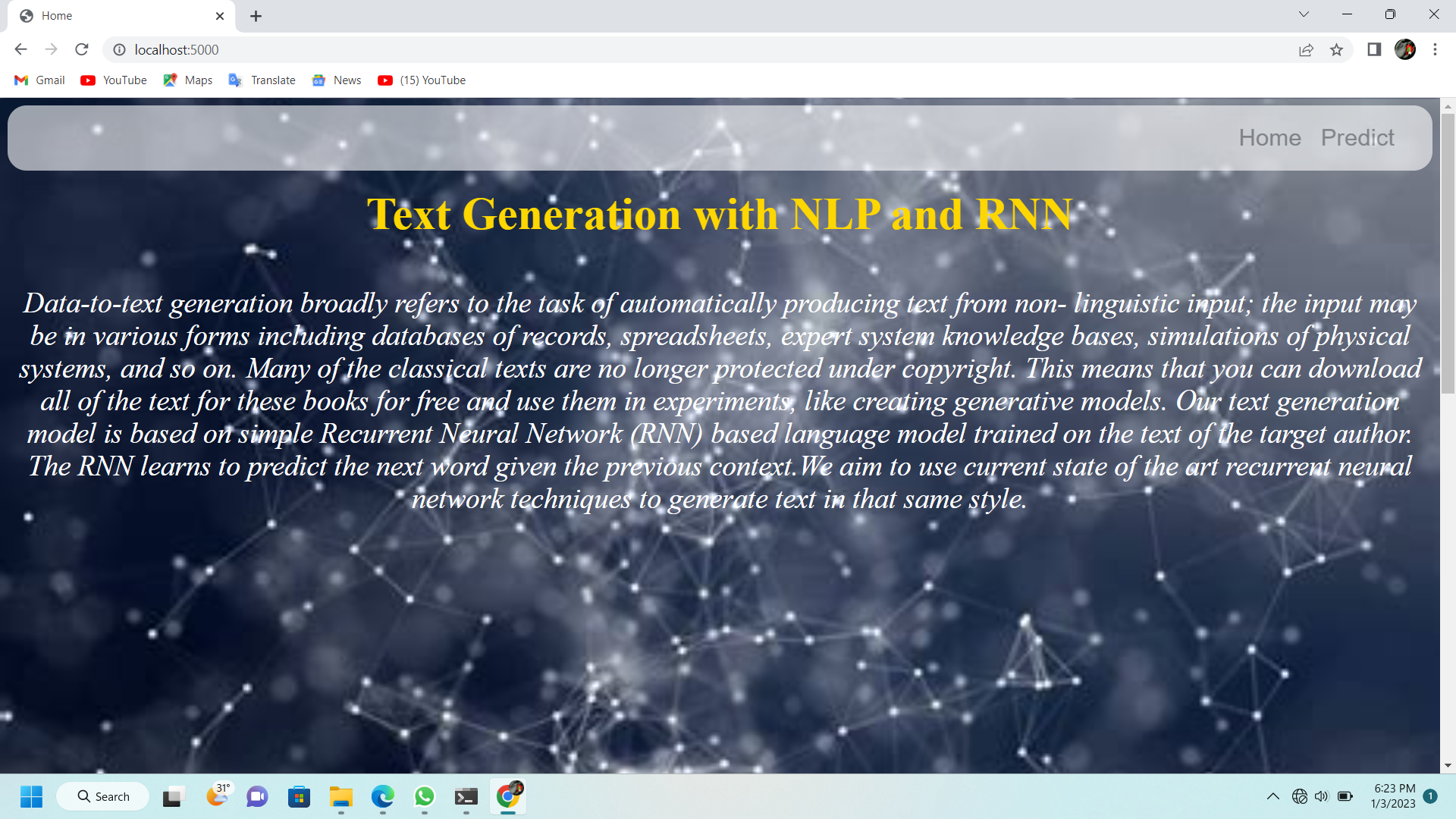


**4.FLOWCHART :**

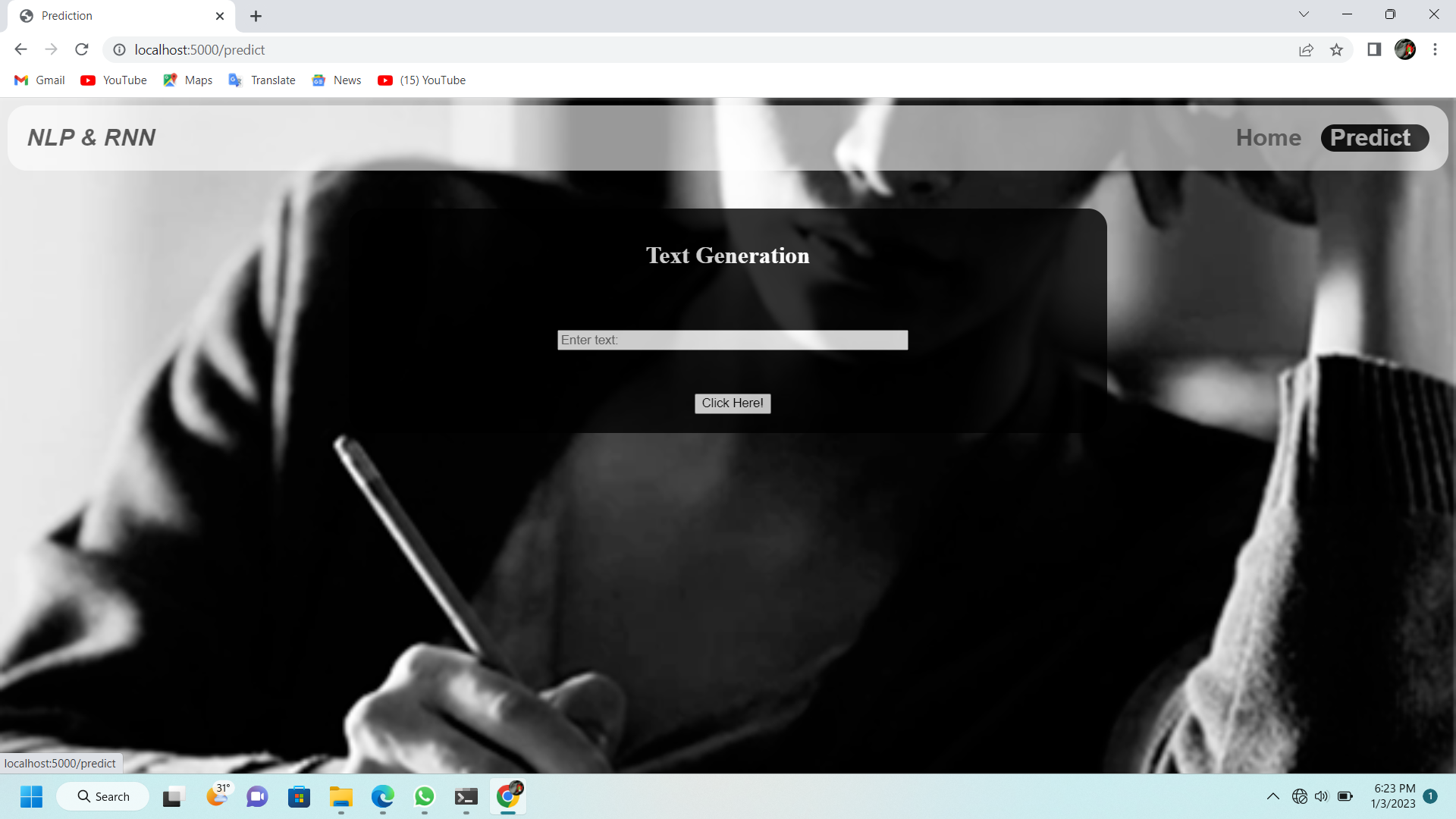


**Fig. : Flow chart of Text Generation using LSTM**

**5 . RESULT :**



**Fig. : Home.HTML**

 **Fig. :Predict.HTML**

**7 . References :**

[1] Li Su. The market approach and reflection of the development of machine journalis- Taking Autamated Insights as an example[J]. press. 2015(18):56-61(in Chinese)

[2] Wan Xiaojun, Feng Yansong, Sun Weiwei. Research Progress and Trend of Automatic Text Generation. CCF Chinese Information Technology Professional Committee (in Chinese)

[3] Brown T B, Mann B, Ryder N, et al. Language Models are Few-Shot Learners[J]. 2020.

[4] Hochreiter, Sepp & Schmidhuber, Jürgen. (1997). Long Short-term Memory. Neural computation. 9. 1735-80. 10.1162/neco.1997.9.8.1735.

[5] Sak, H. & Senior, Andrew & Beaufays, F. (2014). Long short-term memory recurrent neural network architectures for large scale acoustic modeling. Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH. 338-342.

[6] Gers F A, Schmidhuber J. Recurrent nets that time and count[C]// Neural Networks, 2000. IJCNN 2000, Proceedings of the IEEE-INNS-ENNS International Joint Conference on. IEEE, 2000.

[7] Sun Baoshan,Li Wei.Research on Chinese Word Segmentation with Circulatory Network Connected by Peepholes[J]. Computer Engineering and Applications, 2019(19). (in Chinese)

[8] Cho, Kyunghyun & van Merriënboer, Bart & Gulcehre, Caglar & Bougares, Fethi & Schwenk, Holger & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation. 10.3115/v1/D14-1179.

[9] Wiseman S, Rush A M. Sequence-to-Sequence Learning as Beam-Search Optimization[J]. 2016.

[10]Mikolov T, Chen K, Corrado G, et al. Efficient Estimation of Word Representations in Vector Space[J]. Computer ence, 2013.

[11]Pennington J, Socher R, Manning C. Glove: Global Vectors for Word Representation[C]// Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). 2014.

[12]Zhang T, Kishore V, Wu F, et al. BERTScore: Evaluating Text Generation with BERT[J]. 2019.Zhang T, Kishore V, Wu F, et al. BERTScore: Evaluating Text Generation with BERT[J]. 2019.

[13]Sellam T, Das D, Parikh A P. BLEURT: Learning Robust Metrics for Text Generation[J]. 2020.

[14]Peters M, Neumann M, Iyyer M, et al. Deep Contextualized Word Representations[C]// Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers). 2018. [15]Yang Z, Dai Z, Yang Y, et al. XLNet: Generalized Autoregressive Pretraining for Language Understanding [J]. 2019.

[16]Qin Y, Song D, Chen H, et al. A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction[J]. 2017