Project Report on

Title Generator

at

U. V. Patel College of Engineering

Prepared by:

Hetshree Patel (18012011065)

Guide:

Prof. Ketan Sarvakar

B.Tech Semester VII
(Computer Engineering)
December 2021

Submitted to,
Department of Computer Engineering
U.V. Patel College of Engineering
Ganpat University, Kherva - 384 012

Abstract:

Deep learning and Natural language processing (NLP) are growing in popularity for their use in ML technologies like self-driving cars, speech recognition software, text generator, title generator etc. Title generation is popular across the board and in every industry, especially for mobile app like YouTube. Even journalism uses title generation to aid writing processes. It also uses to give unique titles for articles, blogs etc.

Problem Description:

In this case study, I used the YouTube trending videos dataset and the Python programming language to train a model of title generation language using deep learning, which will be used for the task of title generator for YouTube videos or even for your blogs.

Title generator is a natural language processing task and is a central issue for several deep learning, including text synthesis, speech to text, and conversational systems.

Objectives:

- Main aim of title generator is it generate any length of title for your video and blog by providing only single word.
- Title is one of the most important pieces of your video metadata so this model generates best title with appropriate words

Notebook used:

Colab - 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. Colab notebooks allow you to combine executable code and rich text in a single document, along with images, HTML, Latex and more.

Tool, Technology and Library requirements:

- **Pandas**

  pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series.

- **NumPy**

  NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.

- **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.

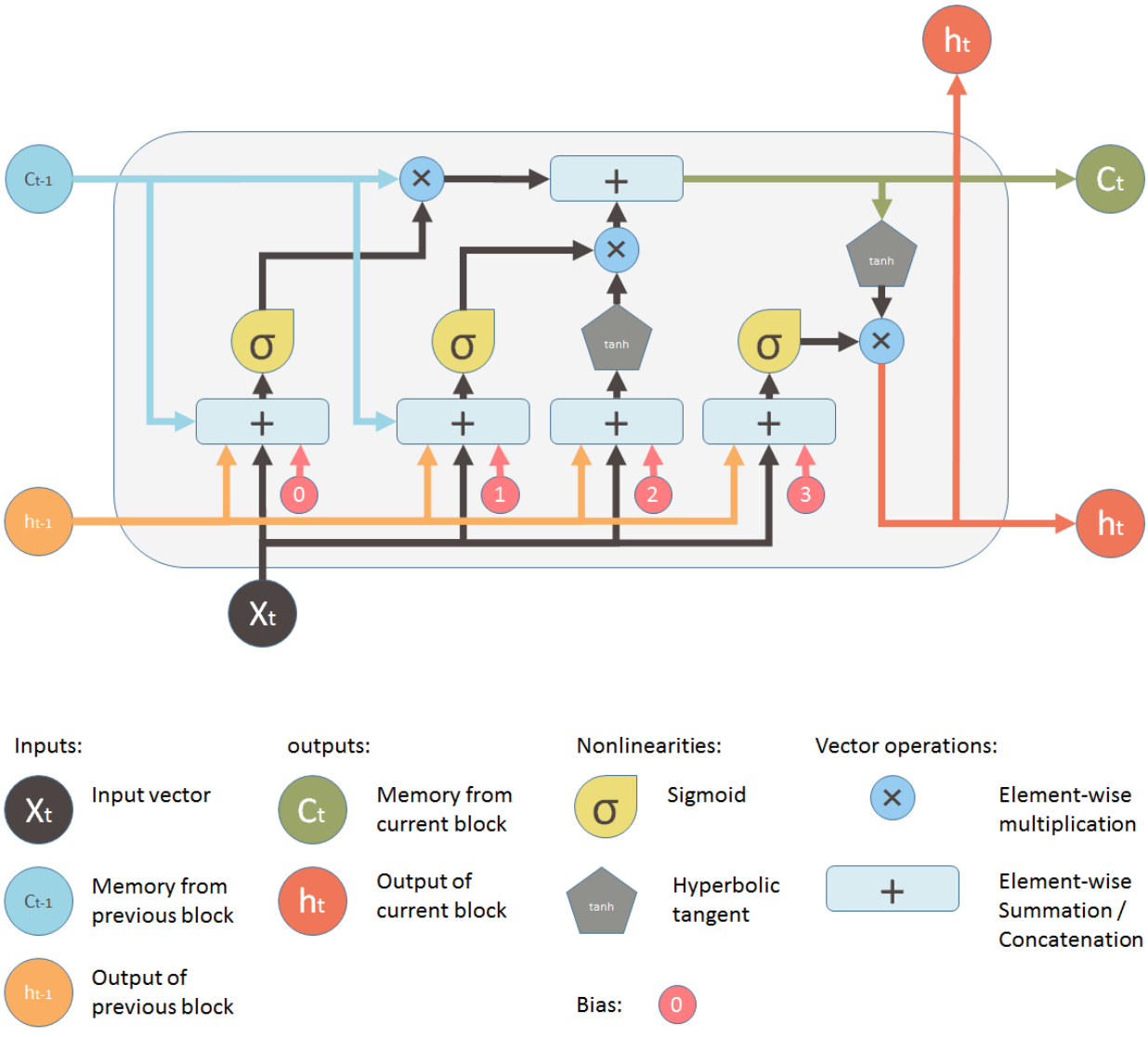
- **TensorFlow**

  TensorFlow is an open-source library developed by Google primarily for deep learning applications. It also supports traditional machine learning. TensorFlow was originally developed for large numerical computations without keeping deep learning in mind.

- **LSTM model**

  LSTM networks were designed specifically to overcome the long-term dependency problem faced by recurrent neural networks RNNs.

Architecture of LSTM:



In recurrent neural networks, the activation outputs are propagated in both directions, i.e. from input to output and outputs to inputs, unlike direct-acting neural networks where outputs d activation are propagated in only one direction. This creates loops in the architecture of the neural network which acts as a “memory state” of neurons.

As a result, the RNN preserves a state through the stages of time or “remembers” what has been learned over time. The state of memory has its advantages, but it also has its disadvantages. The gradient that disappears is one of them.

In this problem, while learning with a large number of layers, it becomes really difficult for the network to learn and adjust the parameters of the previous layers. To solve this problem, a new type of RNN has been developed; LSTM (long-term memory).

Work Flow:

1. First we import all the required libraries such as import pandas as pd, from keras.layers import Embedding, LSTM, Dense, Dropout, import tensorflow as tf and many more.

2. Now we need to process our data so that we can use this data to train our machine learning model for the task of title generator.

3. Generating Sequences

Natural language processing tasks require input data in the form of a sequence of tokens. The first step after data cleansing is to generate a sequence of n-gram tokens.

An n-gram is an adjacent sequence of n elements of a given sample of text or vocal corpus. Elements can be words, syllables, phonemes, letters, or base pairs. In this case, the n-grams are a sequence of words in a corpus of titles. Tokenization is the process of extracting tokens from the corpus.

4. Padding the sequences

5. Title Generator with LSTM Model

The LSTM model contains an additional state (the state of the cell) which essentially allows the network to learn what to store in the long term state, what to delete and what to read.

Layers of LSTM Model

Input layer: takes the sequence of words as input
LSTM Layer: Calculates the output using LSTM units.
Dropout layer: a regularization layer to avoid overfitting
Output layer: calculates the probability of the next possible word on output

6. Testing The Model

7. Generating Titles

Dataset Name:

CAvideos.csv, USvideos.csv and GBvideos.csv

Dataset location:

https://drive.google.com/drive/folders/1LIhV4bW_ITRA2IoY7WwvmKx_OtltLYzL?usp=sharing

Code:

https://github.com/Hetshree1611/Title-Generator/blob/main/Title_Generator.ipynb

Output:

```
[ ] print (generate_text("", 5, lstm_model, max_sequence_len))
    print (generate_text("euclidean", 4, lstm_model, max_sequence_len))
    print (generate_text("generative", 5, lstm_model, max_sequence_len))
    print (generate_text("ground breaking", 5, lstm_model, max_sequence_len))
    print (generate_text("new", 4, lstm_model, max_sequence_len))
    print (generate_text("understanding", 5, lstm_model, max_sequence_len))
    print (generate_text("long short term memory", 6, lstm_model, max_sequence_len))
    print (generate_text("LSTM", 6, lstm_model, max_sequence_len))
    print (generate_text("a", 5, lstm_model, max_sequence_len))
    print (generate_text("anomaly", 5, lstm_model, max_sequence_len))
    print (generate_text("data", 7, lstm_model, max_sequence_len))
    print (generate_text("designing", 7, lstm_model, max_sequence_len))
    print (generate_text("reinforcement", 7, lstm_model, max_sequence_len))
```

```
 React To Try To The
Euclidean React To Try To
Generative React To Try To The
Ground Breaking The Thinks The Webs On
New Panther Of The Last
Understanding React To Try To The
Long Short Term Memory On The Last Jedi The Hd
Lstm React To Try To The Last
A Voice 2018 Blind Hd Or
Anomaly React To Try To The
Data Voice Episode 1 Hum Tv Episode 12
Designing React To Try To The Last Jedi
Reinforcement React To Try To The Last Jedi
```

```
[ ] print (generate_text("Spiderman", 7, lstm_model, max_sequence_len))

    Spiderman The Last Jedi The Hd Of The
```

```
[ ] print (generate_text("HUM", 7, lstm_model, max_sequence_len))

    Hum Episode 23 7Th May 2018 Ary Digital
```